



US005608874A

United States Patent [19]

Ogawa et al.

[11] **Patent Number:** 5,608,874[45] **Date of Patent:** Mar. 4, 1997

[54] **SYSTEM AND METHOD FOR AUTOMATIC DATA FILE FORMAT TRANSLATION AND TRANSMISSION HAVING ADVANCED FEATURES**

[75] **Inventors:** Stuart S. Ogawa, San Francisco, Calif.; Kevin R. Pierce, Fairview, N.J.

[73] **Assignee:** AutoEntry OnLine, Inc., San Jose, Calif.

[21] **Appl. No.:** 435,071

[22] **Filed:** Apr. 8, 1995

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 349,022, Dec. 2, 1994.

[51] **Int. Cl.⁶** G06F 13/00

[52] **U.S. Cl.** 395/200.15; 395/840

[58] **Field of Search** 395/200.01, 200.03, 395/840, 841, 412, 416, 421.11, 182.16, 600, 200.02, 200.09, 200.13, 200.17, 200.18, 853, 650, 824, 830, 831, 839, 200.15; 364/130, 401, 406, 152; 371/31, 32, 67.1, 68.2

[56] **References Cited****U.S. PATENT DOCUMENTS**

4,831,582	5/1989	Miller et al.	395/600
4,951,196	8/1990	Jackson	364/401
5,054,096	10/1991	Beizer	382/41
5,119,465	6/1992	Jack et al.	395/500
5,131,053	7/1992	Bernzott et al.	382/176
5,257,369	10/1993	Skeen et al.	395/650
5,299,304	3/1994	Williams et al.	395/148
5,317,740	5/1994	Sites	395/700
5,416,917	5/1995	Adair et al.	395/500
5,473,691	12/1995	Menczes et al.	380/25
5,491,473	2/1996	Gilbert	340/870.01
5,507,030	4/1996	Sites	395/800
5,513,323	4/1996	Williams et al.	395/200.18

OTHER PUBLICATIONS

The Forms Processing Paradigm Shift, A. Moore, Imaging Magazine, Mar. 1995, pp. 84-95.

Forms Processing Sampler, A. Moore, Imaging Magazine, Mar. 1995, pp. 96-106.

Teleform for Windows—Fast Simple & Easy.

Cornerstone—New Technology Reads "Impossible" Forms. GTESS Corp.—Automated Data Entry System, Speed the Entry of Vital Data While Reducing Costs by 50 Percent or More.

Primary Examiner—Alpesh M. Shah

Assistant Examiner—Dhiren Rana Odedra

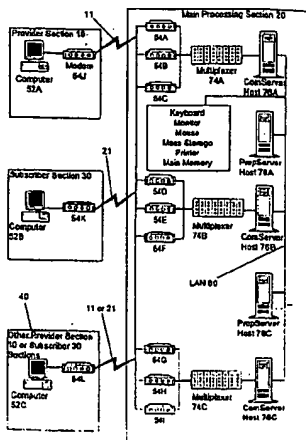
Attorney, Agent, or Firm—R. Michael Ananian; Flehr, Hohbach, Test, Albritton & Herbert

[57]

ABSTRACT

Method, system, and apparatus are described for automatically receiving, at an intermediate processing location, data from a wide variety of remote sources, identifying the format of the data, translating the data to a common file format, sending the data to a recipient in an intermediate format, then translating the data to the specific format needed by the particular recipient. The system operates automatically with little human intervention. A unique system for automatically selecting and implementing specific translation modules is also described. Error checking features ensure that the transferred data matches the original data although the format is altered, and documentary receipts are sent to each section of the system that sends data, and logical, statistical and mathematical operations may be performed on the data. The system utilizes internal databases which allows it to know what format data will arrive in, what format to translate it to, and how many transactions to bill a data-receiving subscriber for. The system performs data translation and transfer, and performs validation, exception reporting, data analysis, and generates and sends receipts. Some or all such data processing may alternatively be performed at one or both of the data source and the data destination, without an intermediate processing location. A provider data file may also be prepared from a scanner, such as a fax transmission, which provides image data which are processed in order to derive the data file.

28 Claims, 16 Drawing Sheets



Error Detection
Bad Transmission

Figure 1

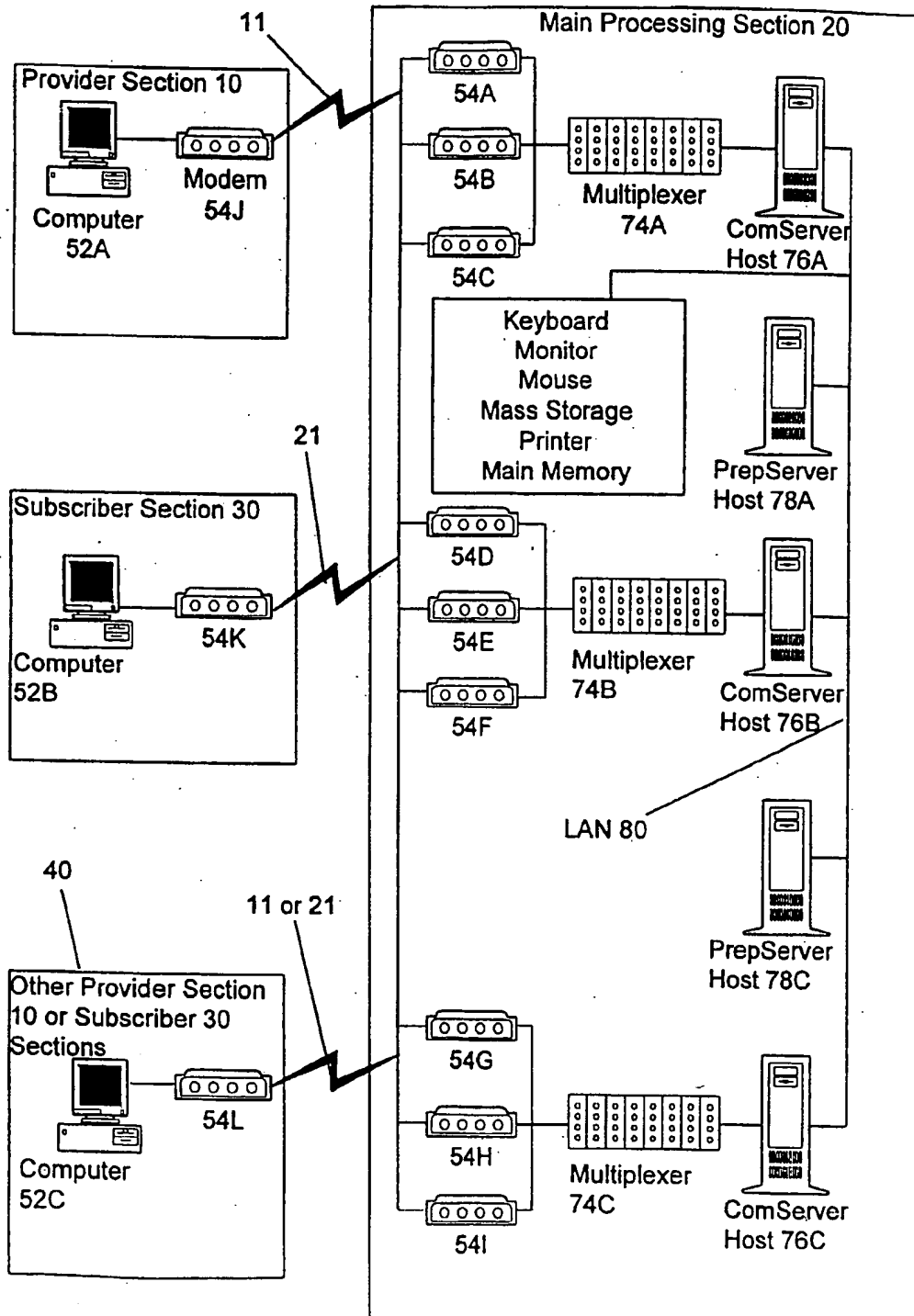
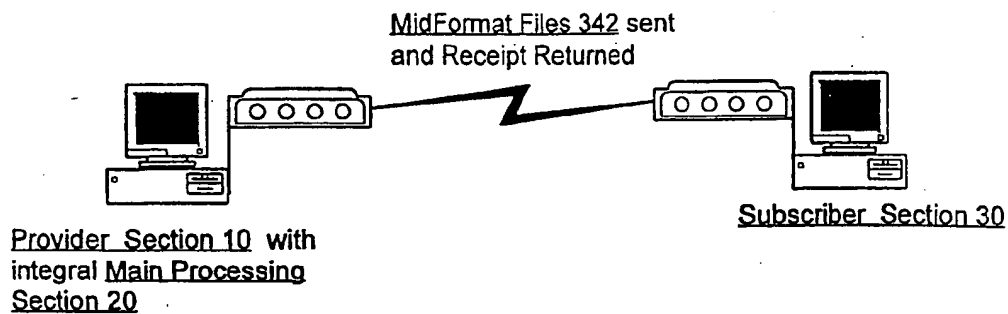
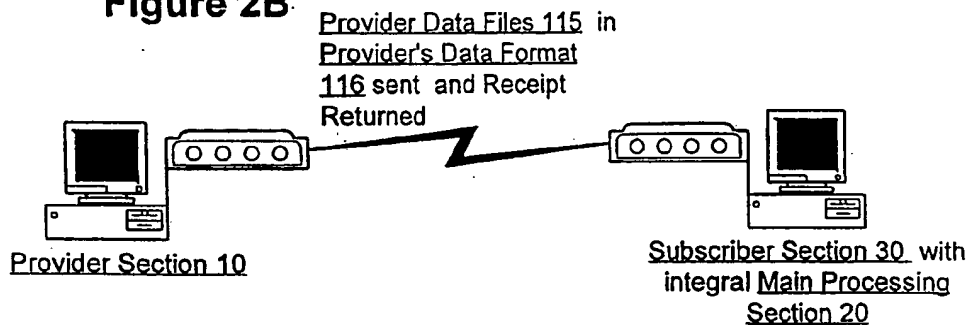
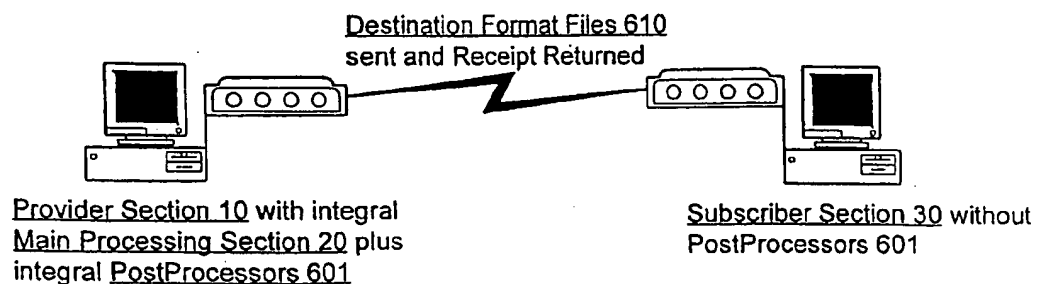


Figure 2A**Figure 2B****Figure 2C**

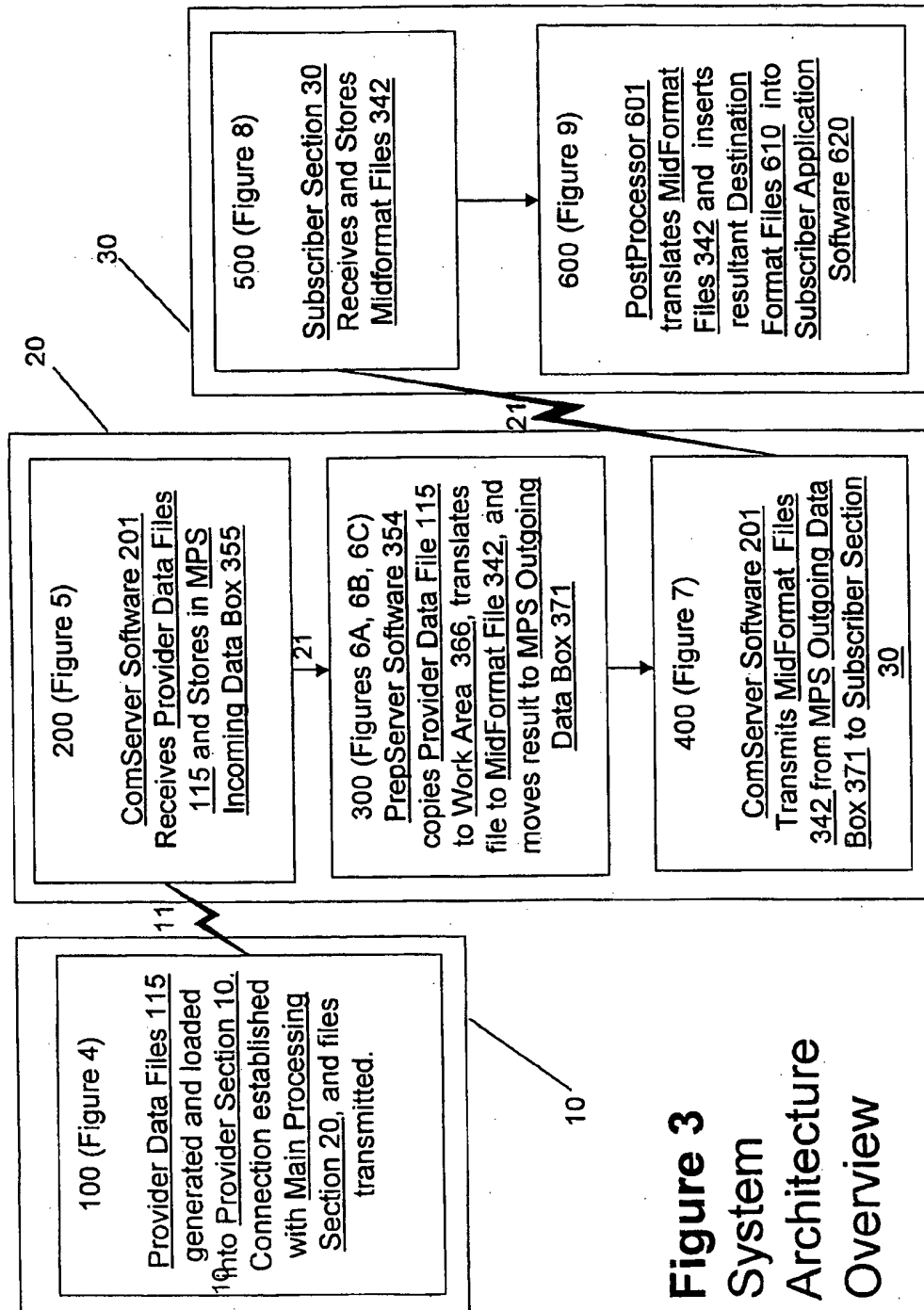


Figure 4: Details of Block 100

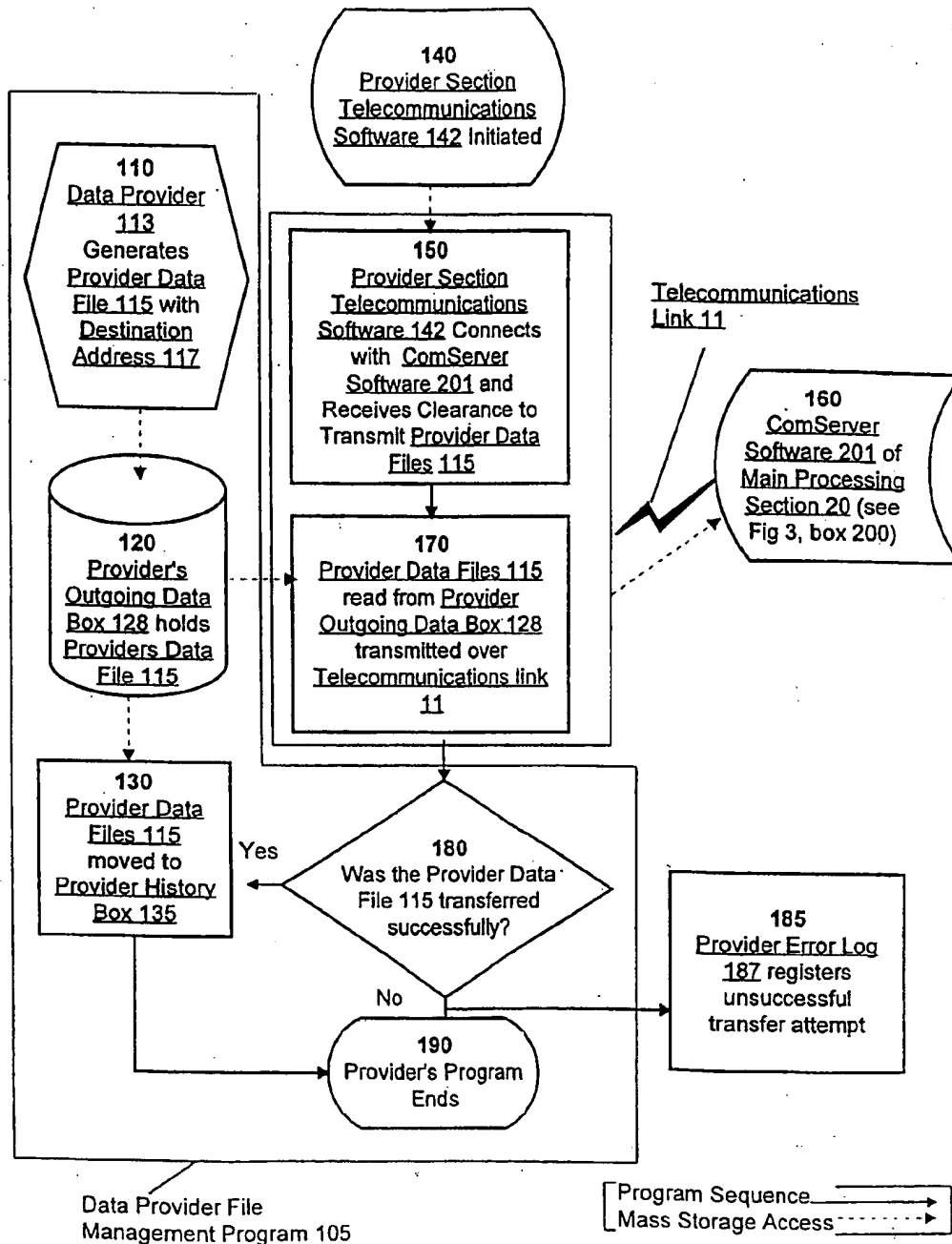


FIGURE 5A: Detail of Block 200

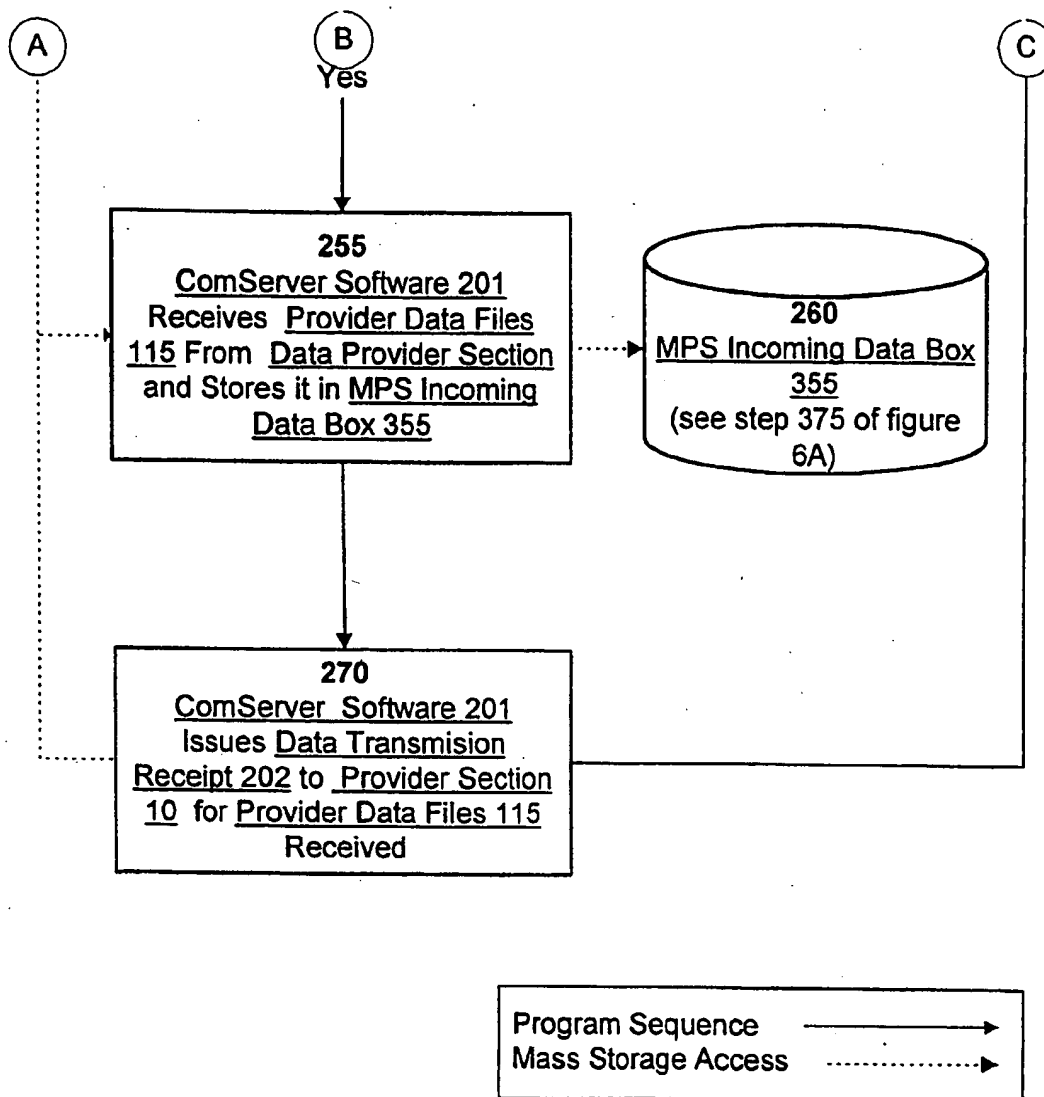


FIGURE 5B: Detail of Block 200

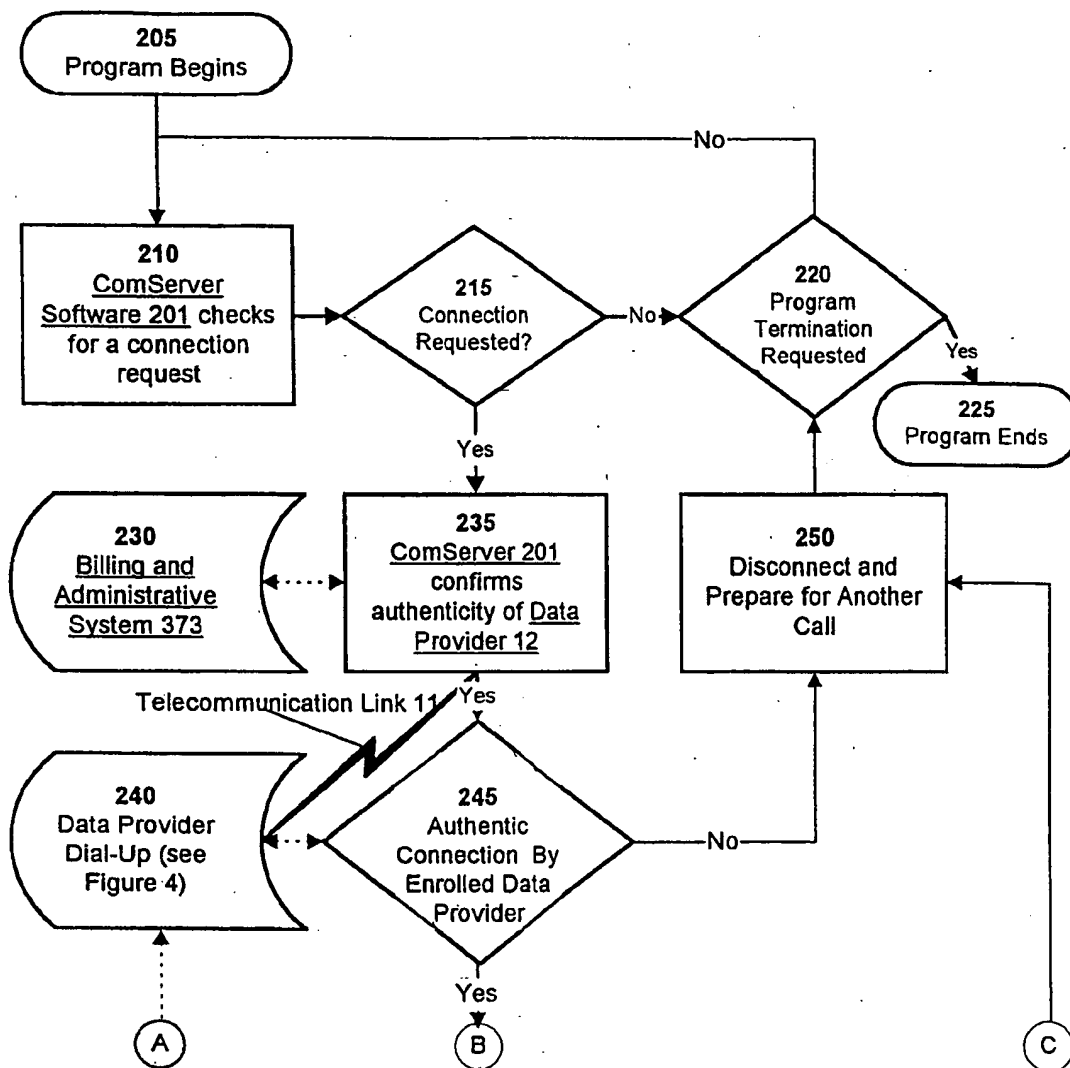
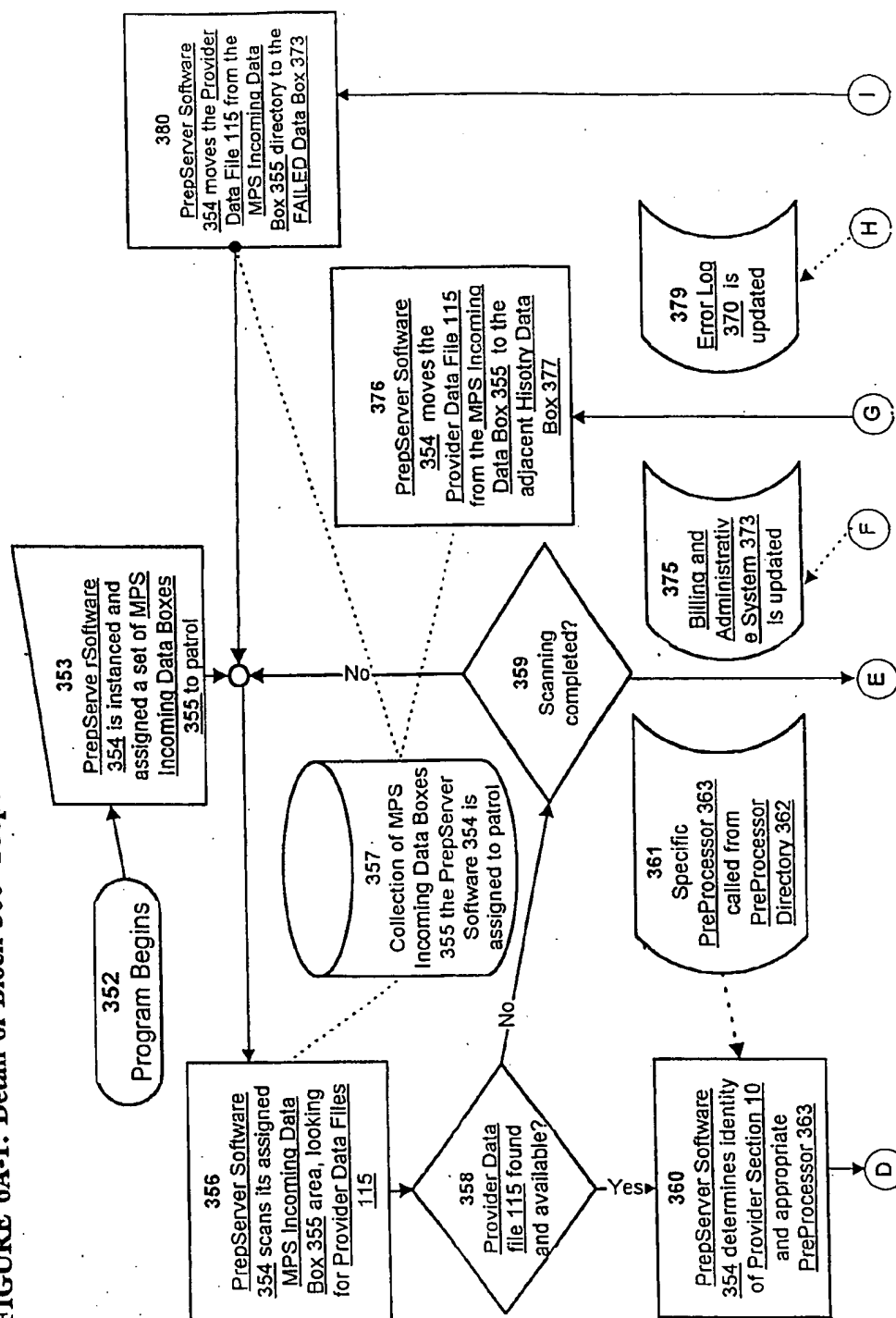


FIGURE 6A-1: Detail of Block 300- PrepServer Software Function



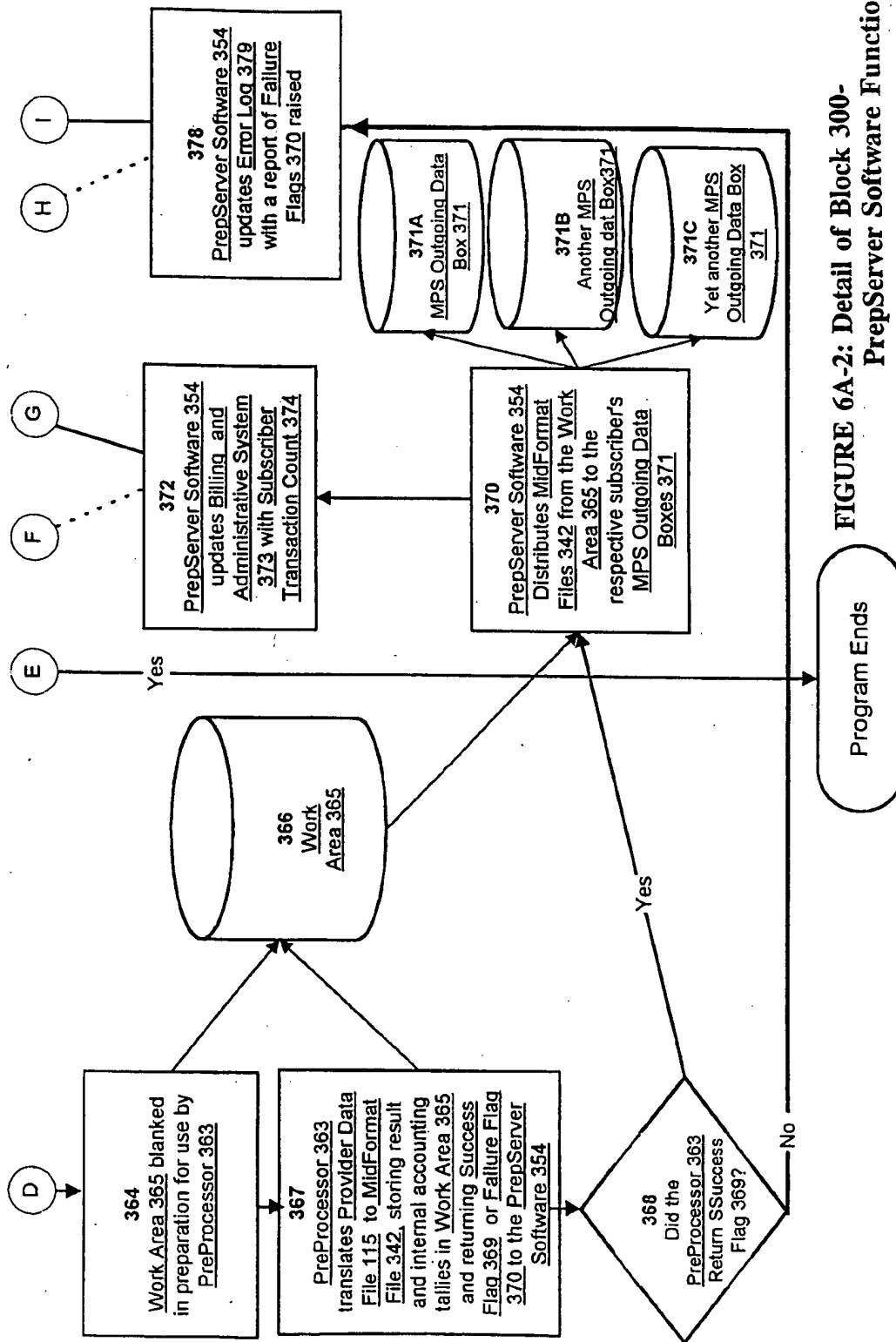


FIGURE 6A-2: Detail of Block 300-
PrepServer Software Function

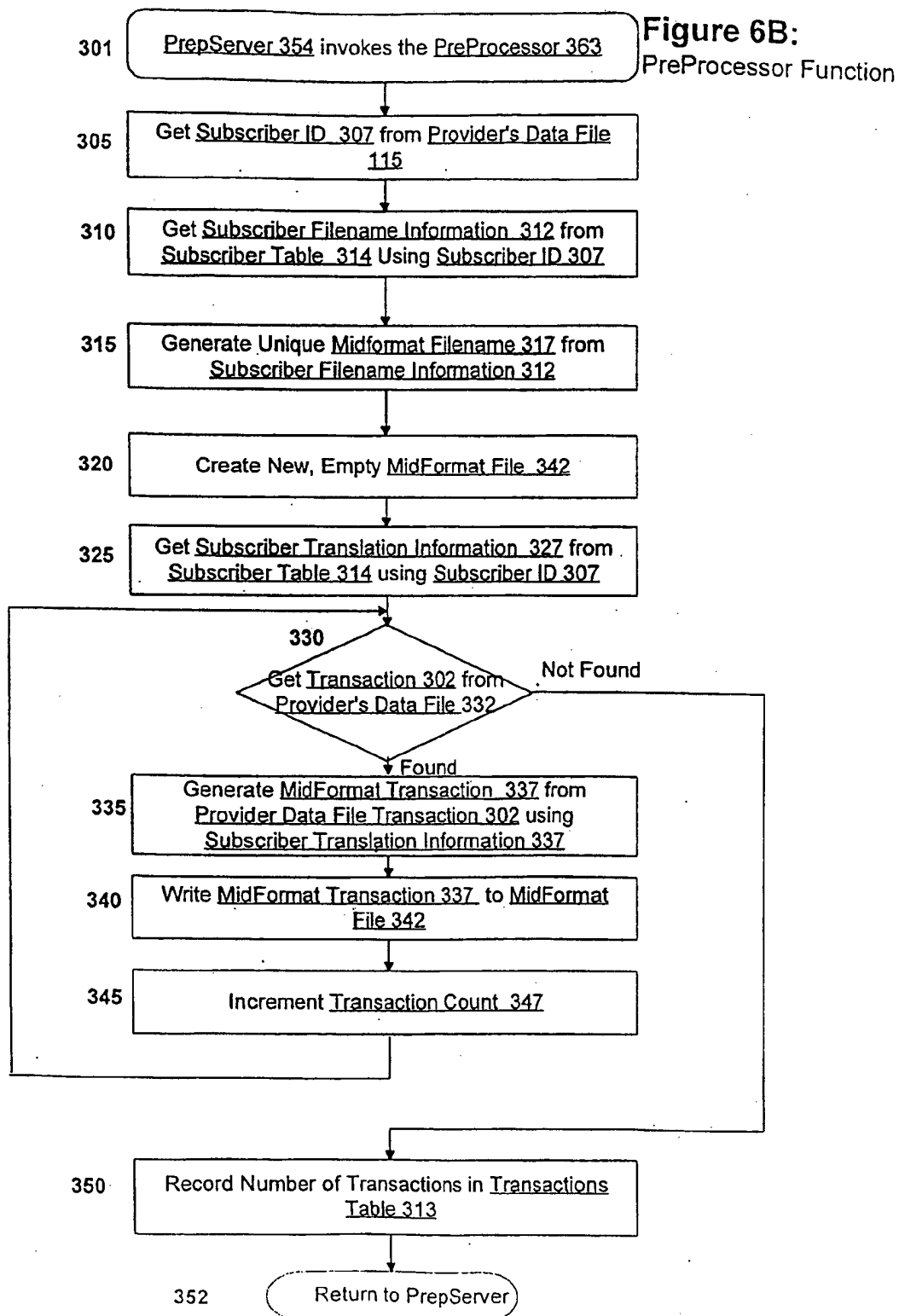
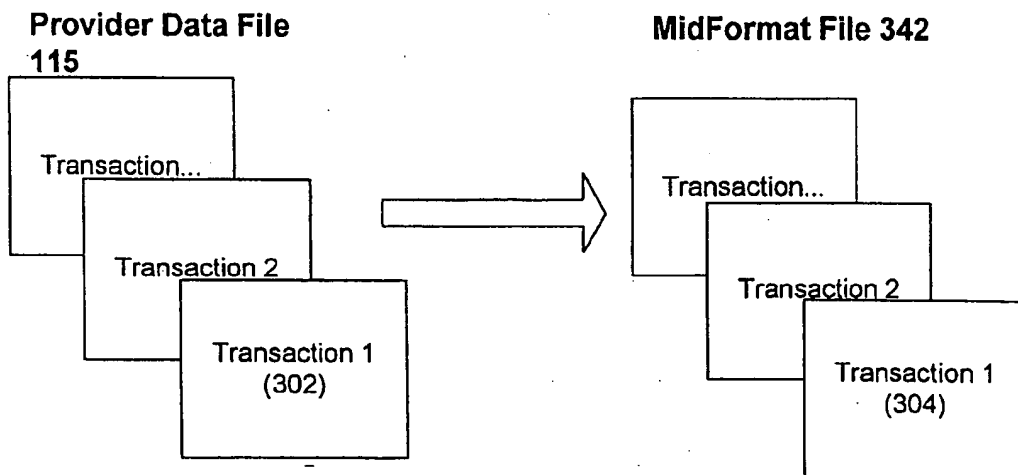


FIGURE 6C-1



Subscriber Table 314 Examples

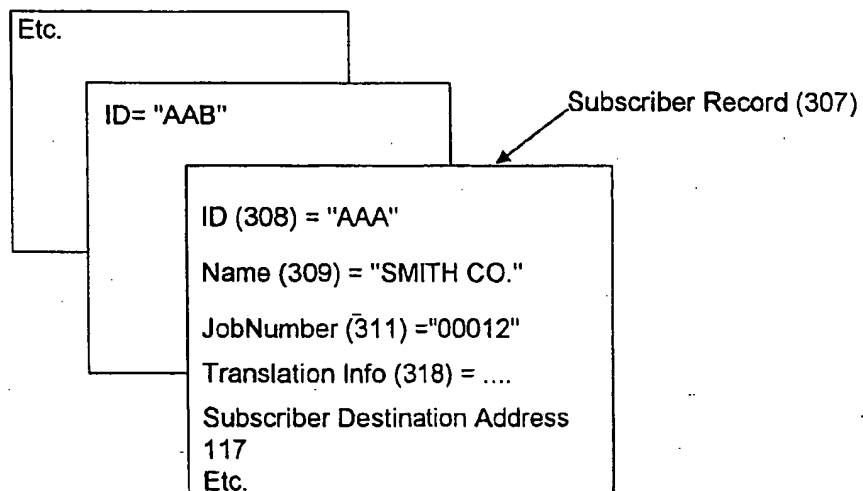


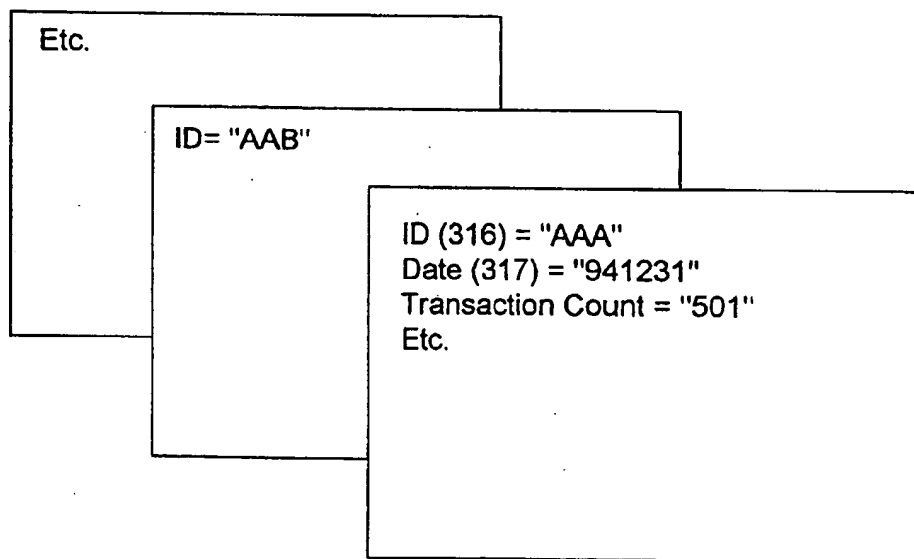
FIGURE 6C-2**Transactions Table (313) Examples**

FIGURE 7A: Detail Block 400

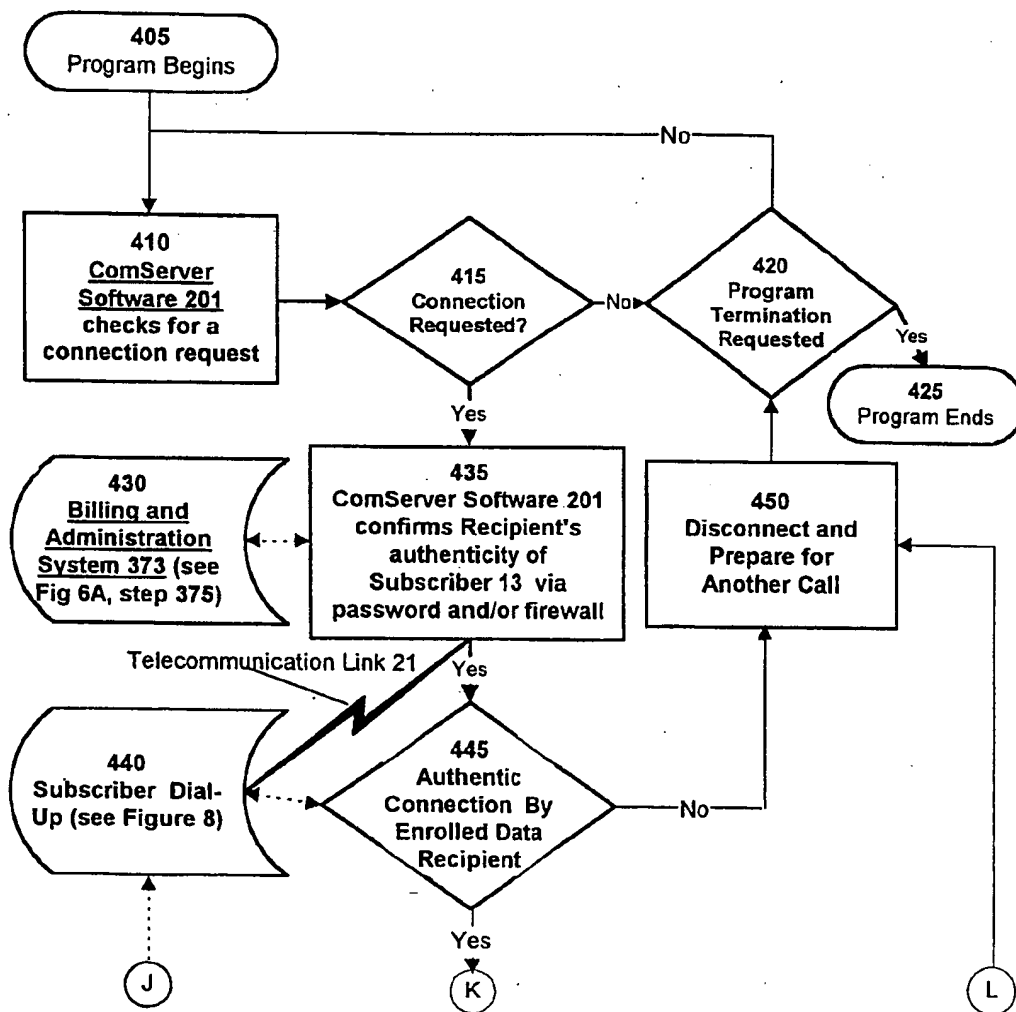


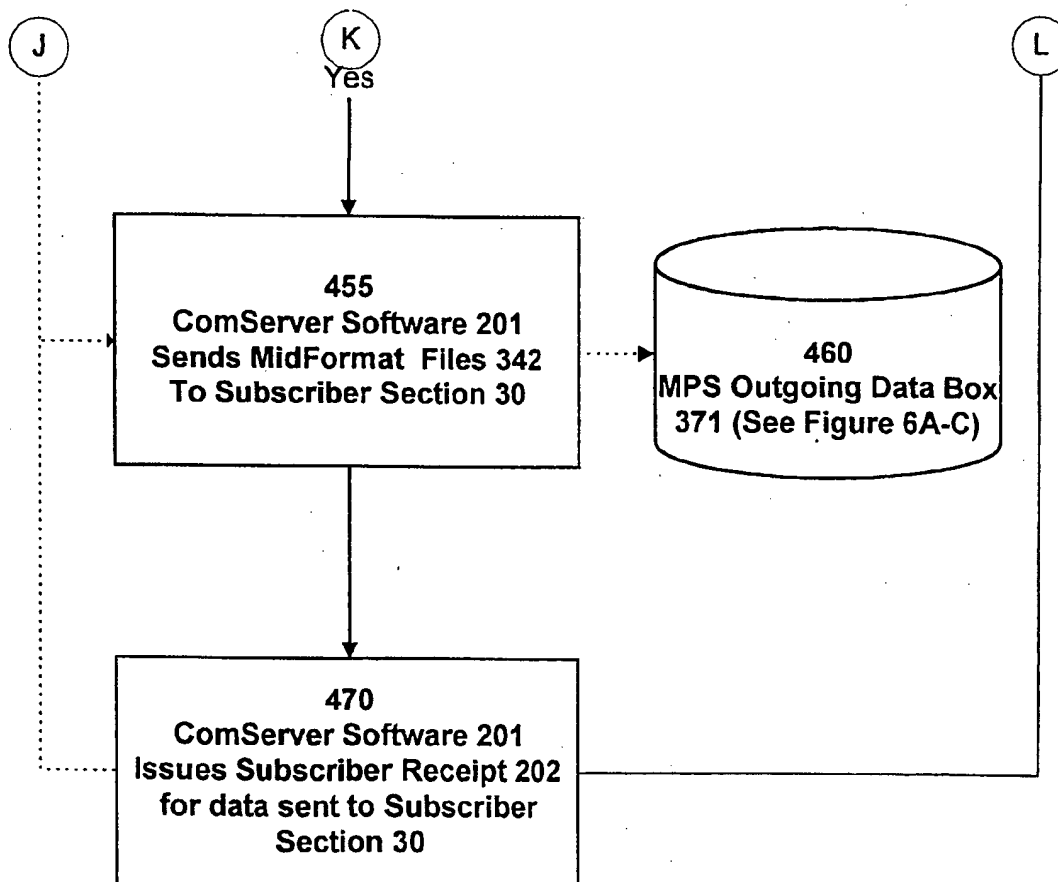
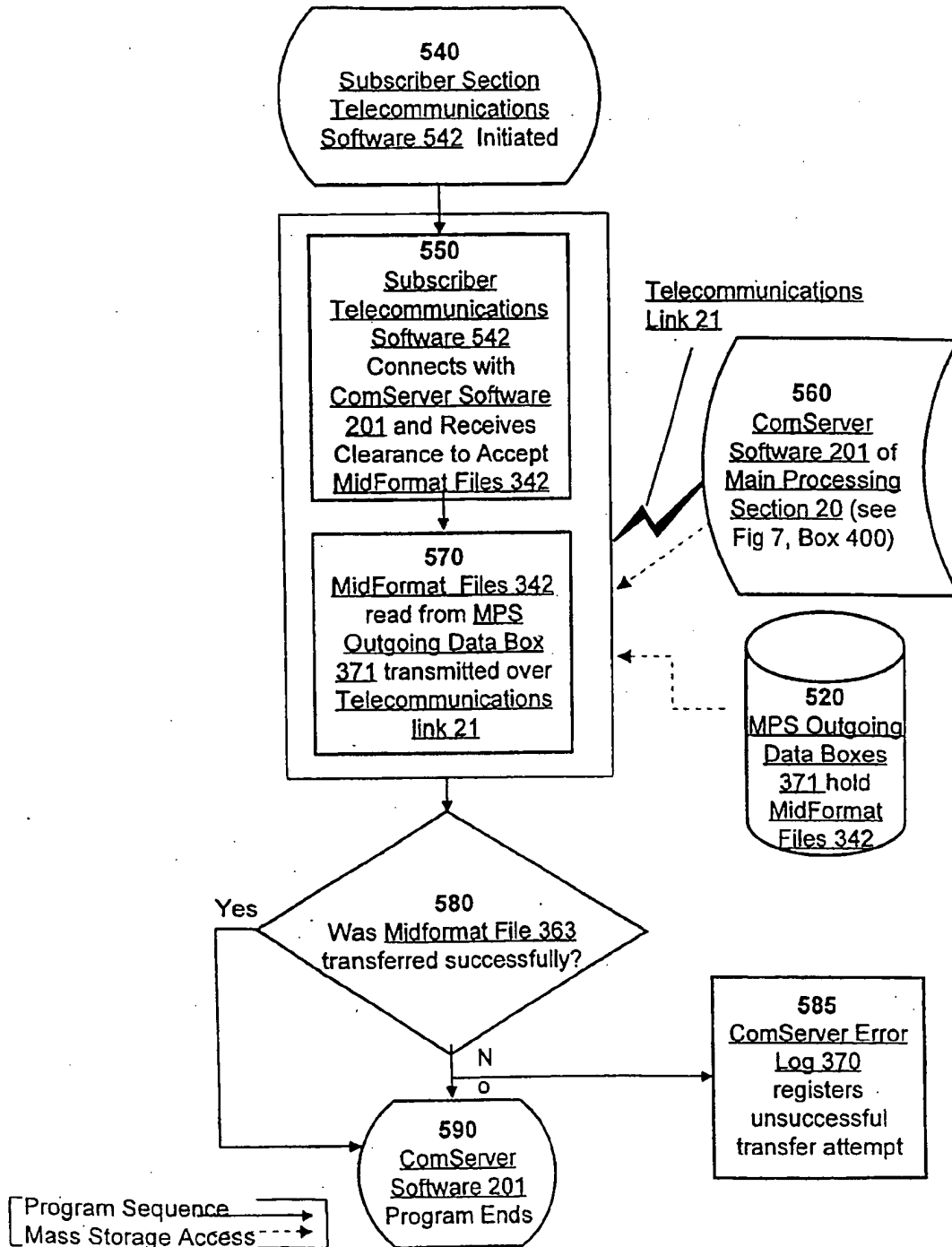
FIGURE 7B: Detail of Block 400

Figure 8: Details of Block 500



**Figure 9: Detail of
Block 600**

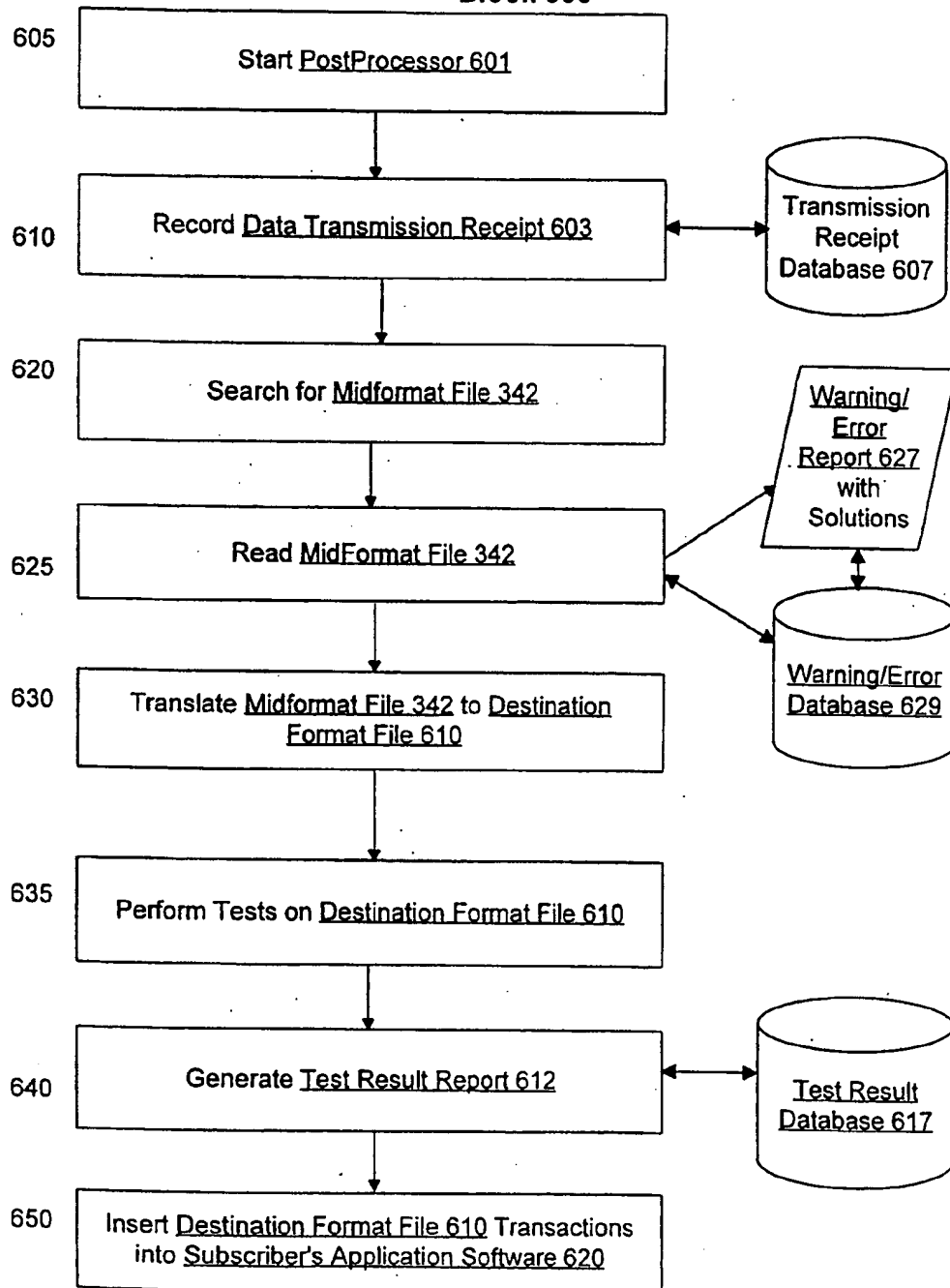
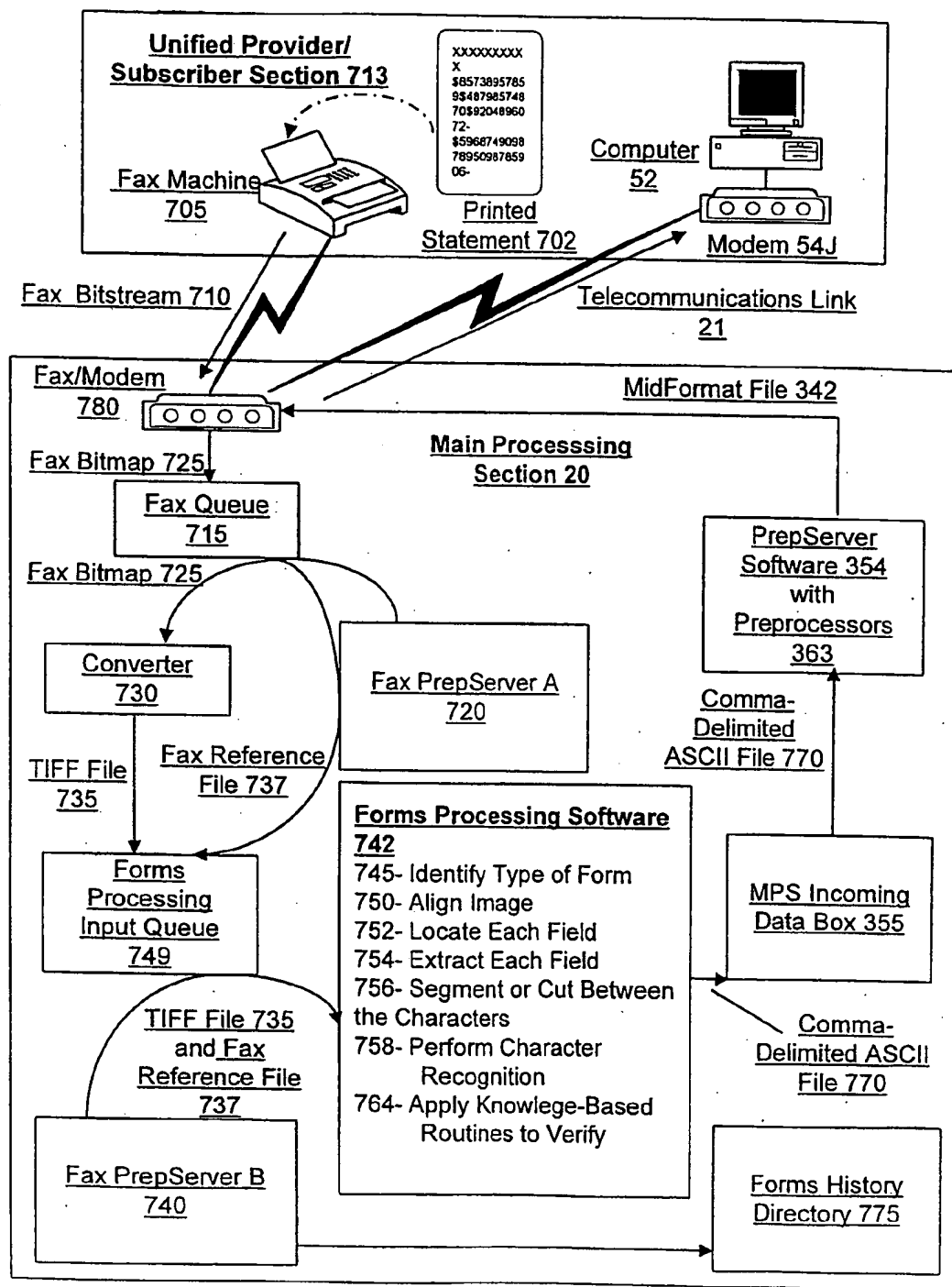


FIGURE 10



SYSTEM AND METHOD FOR AUTOMATIC DATA FILE FORMAT TRANSLATION AND TRANSMISSION HAVING ADVANCED FEATURES

RELATED APPLICATIONS

This application is a continuation-in-part of U.S. application Ser. No. 08/349,022 filed Dec. 2, 1994.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The field of the present invention relates to automated data transfer and translation. The invention provides a method and structure for automatically transferring data, such as accounting data, from a variety of remote sources, in a variety of formats to a plurality of remote sites, where the data is entered into computer applications or storage, the transfer including format translation for compatibility. The invention serves as means to eliminate time-consuming and error-producing manual data re-entry.

2. Background

Computerized data, or electronically stored information, must frequently be moved from one computer to another. In the case of accounting data, for example, banks generally keep computerized records of all of transactions affecting their client's accounts—information which the clients often also wish to keep track of on their own computerized accounting systems. In response to this, a variety of sophisticated computer accounting programs have recently become available to users of small and medium powered computers. However, due to a variety of reasons discussed below, such clients usually have to re-enter data manually from paper printouts obtained from the data provider, for example, from statements from a bank. Manual re-entry of data is not only time-consuming, and hence expensive to undertake in terms of man-hours, but it also is likely to introduce errors into the data set. It has been estimated that manual data re-entry, verification, and validation costs several dollars per transaction.

Methods of automated data transfer are known in the art. The "Intellicharge" system for Quicken (Intuit Inc., Menlo Park, Calif.) downloads credit card transaction information to a Quicken user's computer. The data used, however, comes from a uniform source—a single bank. Hence, Intellicharge does not employ a multiple-format data translation scheme, nor a multiple-source data transmission scheme. Similarly, in the United States Internal Revenue Service "Electronic Filing" program, data is entered and transmitted in a single, specific pre-prescribed format, to a single recipient.

Methods of data translation are also known in the art. A software application entitled "Data Junction" (Tools and Techniques, Inc., Austin, Tex.) translates multiple formats of data. However, the package depends on manual operation to designate the files to be translated, and the formats of the source and destination files. Furthermore, this software does not perform any data transfer, verification, validation, exception reporting, or journal entry correction.

Conventional electronic data exchange (EDI) systems involve two or more companies that have agreed to interact with one another according to a pre-designated standard dictated by the industry in which the transaction is taking place. In order for such a system to work for a given industry, there must be an agreed-upon standard that is

used—much like in the case of the IRS system described above. Those industries that do not have such a standard cannot participate. Data analysis, such as exception reporting or statistical analysis, are not features of such systems. Obviously, such systems lack flexibility and versatility. Additionally, the computer systems that support conventional EDI are expensive to operate and maintain because they are specialized to serve specific industry segments, and hence cannot achieve the efficiency and low cost that economies of scale might permit in a more widely applicable system.

In summary, conventional methods of automated data transfer, and of data translation, are quite limited, due primarily to limited network transfer capabilities, and the lack of universal data format standards. Hence, anyone wishing to automatically transfer data from a variety of computer systems to a variety of others must contend with a plethora of incompatible formats, and a lack of reliable transfer and error detection means. For these reasons, existing data transfer systems have been tailored to work with only one, or very few types of data sources and recipients, and these data translation methods rely heavily on manual intervention. Data transfer technologies and data translation technologies have not, in the prior art, been efficiently integrated.

OBJECTS AND SUMMARY OF THE INVENTION

As can be appreciated from the foregoing discussion, it would be desirable to have a flexible transfer and translation system that would operate on a wide variety of formats, not hampered by the lack of data format standards—a system that takes data in any format, transfers it to a remote location, and inserts it into the software application in which the data is needed. It would be desirable for a system to automatically understand what specific format translations are needed for a specific data recipient, then to be able to automatically carry out such a translation regardless of what format the data was originally in. It would also be desirable for such a system to operate with minimal user interaction, making it faster, cheaper, and more reliable than manual or semiautomatic performance of such tasks. It would also be desirable to enable companies that want to make their computerized data available to outside agencies and individuals to be able to export this data without the time and expense of developing a custom export system. It would also be desirable to have such a system that is modular and scalable on hardware and software levels, so as to provide reliability, serviceability, and adjustability of the system's performance. The present invention fills these aforementioned needs.

The present invention, in one embodiment, comprises a system capable of automatically receiving, at an intermediate processing location, data from a wide variety of remote sources, identifying the format of the data, translating the data to a common format, sending the data to a recipient in an intermediate format, then translating the data to the specific format needed by the particular recipient. A unique system for automatically selecting and implementing specific translation modules may be used. Error checking features ensure that the transferred data matches the original data, and receipts are sent to each section of the system that sends data, and logical, statistical and mathematical operations may be performed on the data. The system utilizes internal databases which allows it to know what format data will arrive in, what format to translate it to, and how many

transactions to bill a data-receiving subscriber for. The system performs data translation and transfer, and performs validation, exception reporting, data analysis, and generates and sends receipts. In alternative embodiments, some or all such data processing is performed at one or both of the data source and the data destination, without an intermediate processing location.

In an alternative embodiment, a provider data file is prepared from a scanner, such as a fax transmission, which provides image data which are processed in order to derive the data file.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates the physical apparatus of one embodiment of the invention.

FIGS. 2A, 2B, and 2C illustrate the major aspects of the physical apparatus and spacial layout of some alternative embodiments.

FIG. 3 shows an overview of an embodiment of the system architecture from the standpoint of process. Each block within this figure corresponds with one or more detailed separate figures.

FIG. 4 shows one embodiment of the major components and processes which occur at block 100 of FIG. 3.

FIG. 5 shows one embodiment of the major components and processes which occur at block 200 of FIG. 3.

FIGS. 6A, 6B, and 6C show some embodiments of the major components and processes which occur at block 300 of FIG. 3. FIG. 6A focuses on the role of the PrepServer Software. FIG. 6B focuses on the role of a selected PreProcessor. FIG. 6C illustrates some of the components utilized in FIG. 6B.

FIG. 7 shows one embodiment of the major components and processes which occur at block 400 of FIG. 3.

FIG. 8 shows one embodiment of the major components and processes which occur at block 500 of FIG. 3.

FIG. 9 shows one embodiment of the major components and processes which occur at block 600 of FIG. 3.

FIG. 10 shows one embodiment of this invention in which a scanner, such as a fax machine, is used to provide data from a printed document.

BRIEF DESCRIPTION OF THE TABLES

Table 1 shows one embodiment of the Subscriber Database Key to the Master Customer Definition Table.

Table 2 shows one embodiment of the Subscriber Database Key to the Master Mailbox Definition Table.

Table 3 shows one embodiment of the Subscriber Database Key to the Transactions Table for Tracking and Accounting.

Table 4 shows one embodiment of the Subscriber Database Key to the System Log File.

Table 5 shows one embodiment of the Subscriber Database Key to the Incrementing Batch ID Number.

Table 6 shows sample PreProcessor Command-Line Arguments.

Table 7 shows an example of the layout of a payroll file received from a Data Provider.

Table 8 shows an example of the layout of a bank account file received from a Data Provider.

Table 9 shows an example of the layout of a Midformat payroll file.

Table 10 shows an example of the layout of a Midformat bank account file.

Table 11 shows an example of the layout of a Destination Format payroll file.

Table 12 shows an example of the layout of a Destination Format bank account file.

DETAILED DESCRIPTION

Definitions and Terminology

ComServer Host—A machine on which the ComServer Software is operating, allowing the machine to control the telecommunication operations of a networked, distributed-task version of the Main Processing Section.

ComServer Software—Telecommunications software, in some embodiments residing on a ComServer Host computer, which receives remotely transmitted data and stores it. It may also transmit processed and unprocessed data to remote sites. Such a system is similar to a computerized mail box system as commonly seen in on-line services such as America Online (Vienna, Va.). ComServer Software functions may be carried out by commercially available software packages such as WildCat! BBS (Mustang Software, Bakersfield, Calif.)

Data Provider File Management Program—A system which enables a user of a Provider Section to connect to the Main Processing Section, and to transmit Provider Data Files, store copies of data files that have been sent, and which stores data transmission receipts.

Data Validation—Comparison of a data set and a reference data set, including calculated or otherwise derived data, to assess the correctness of the data content of a file. An example is making sure that a specific debit was for a certain amount of money.

Data Verification—Comparison of a data set with a reference data set or value to assess the completeness of the data set's structure. An example is screening for a checksum error.

Destination Address—Information that tells the system to whom a data file is to be sent, and therefore, it which Outgoing Data Box it is to be placed.

Destination Format—The data format required or requested by the Subscriber. PostProcessors translate Mid-Format Files into Destination Format files, and load them into the Subscriber's application software.

Environment Directory—The directory where initialization files, system logs, etc. are kept.

Exception Report—The user report which displays errors or warnings related to transactions that have been processed.

Failed Data Box—A subdirectory used to contain files that were not successfully processed or transmitted at a given location in the system.

Firewall—Security methods which protect a computer system from access, using hardware or software measures. A firewall may be a barrier set up to keep designated Local Area Network (LAN) traffic within an/or without a specific LAN. For example, a firewall can be set up to stop internal e-mail from leaving a LAN, or can be set up to keep outside users from connecting to computers within a LAN.

History Data Box—A subdirectory used to contain files that have been successfully processed or transmitted at a given location in the system.

Mail Box Database—The Main Processing Section database that contains records of Incoming and Outgoing Data Boxes, and to what party they pertain.

Main Processing Section (MPS)—Portions of the invention herein described that are responsible for tasks such as data translation, system usage monitoring, and exception reporting. In one embodiment, the Main Processing Section, including PrepServer Software and/or its PreProcessors and/or the ComServer Software, and/or Billing and Administrative System resides at a site separate from the sites of the Provider Section and the Subscriber. In one such embodiment, data received from the Provider is processed by the Main Processing Section to provide translated data. This translated data is then sent on to a Subscriber who translates the data again. In alternative embodiments, the Main Processing Section and/or some or all of its components may reside on either the Provider or Subscriber computers, or split between them, thus eliminating the intermediate site, or reducing the amount of processing that takes place at such an intermediate site.

MPS Incoming Data Box—The subdirectory in which Provider Data Files are placed when it arrives from a specific provider.

MPS Outgoing Data Box—The subscriber-specific subdirectory in which MidFormat Files are placed when PrepServer has successfully processed the file.

MidFormat File—A data file that has been translated into a format common to all data files of that particular type, prior to being converted to a Destination Format. It originates as a translated Provider Data File created by the PreProcessor, and it contains a list of delimited transactions. This file is translated by the PostProcessor into a Destination Format in order to load it into a Subscriber's application software.

Number of Transactions—A tally internal to each PreProcessor which counts the number of transactions that are processed.

PostProcessor—A program which translates a MidFormat File into the specific Destination Format required by a Subscriber, and loads the Destination Format data into Subscriber's Application Software.

PreProcessors—Software programs controlled by the PrepServer Software, which automatically perform data format translations for specific Data Providers. These programs translate Provider Data Files into MidFormat Files, and record the number of transactions processed into a Transactions Table.

PrepServer Host—A machine in which PrepServer Software is the principal software, allowing the machine to specialize in the PrepServer Software tasks of a networked, distributed version of the Main Processing Section.

PrepServer Software—Software, in some embodiments residing on a PrepServer Host computer, which takes unprocessed data from the ComServer Software by scanning Incoming Data Boxes for received Provider Data Files. If found, the PrepServer Software processes the data with a specific set of instructions (for example, by selecting the appropriate PreProcessor), and places the data in a specific MPS Outgoing Data Box mapped to the data's ultimate destination.

Provider—A Data Provider. A company which sends transaction data to the Main Processing Section. An example is a bank which electronically transfers account statements. Another example is a laboratory which electronically transfers the results of its blood sample analysis.

Provider Data File—A file uploaded into a mailbox by a Provider containing a list of transactions in the Provider's

data format. This file will be translated by the PreProcessor into a MidFormat File. Also known as a Raw Transactions File (RTF).

Provider History Box—The subdirectory in which a Provider Section retains a copy of files that have been successfully transferred to the Main Processing Section.

Provider Outgoing Data Box—The subdirectory in which a Provider Section places Provider Data Files that are ready for transmission.

Provider Section—That hardware and software portion of the system herein described which reside at physical location of the Data Provider, usually including telecommunications software and Provider Data Files to be transferred.

Raw Transactions File (RTF)—See Provider Data File.

Subscriber—Also known as Data Subscriber. The end-recipient of the data in the herein described system. A Subscriber has Subscriber's Application Software (such as a financial software package) which can make use only of data files that are in certain formats. The subscriber wishes to have data residing at the Provider Section to be sent to him, and to be translated in a format that is suitable for use by his application software.

Subscriber's Application Software—A computer application program, such as accounting packages like Platinum (Platinum Software, Irvine, Calif.) into which a Subscriber wishes to enter data. Subscriber's Application software typically requires data to be in a specific format before it can be accepted.

Subscriber ID—The unique identification key assigned to each subscriber.

Subscriber Incoming Data Box—The Subscriber Section subdirectory in which MidFormat files transmitted from the Main Processing Section first arrive, and where they can be found by the PostProcessor.

Subscriber Section—Those hardware and software portions of the system herein described that reside at the physical location of the Data Subscriber, usually including a PostProcessor program.

Subscriber Table—A database file containing subscriber information such as Subscriber ID number, Subscriber Name, Job Number, Translation Information, Subscriber Destination Address, and subscriber Filename Information.

Syslog—A file, such as one in the environment directory, used to record system activity.

Table—Depending upon the context, this term may refer to a display of information used as an exhibit for this patent application, or may be used to mean a database file.

Test Result Report—A report for the user of a subscriber Section displaying errors and warnings related to user specific transactions. Statistical and Boolean logic data validation methods may constitute the basis of the report.

Test Result Database—A Database storing the validation, statistical, and Boolean calculations and algorithms necessary to produce a Test Result Report.

Transaction—Data representing a specific activity, entity, or portion thereof. Examples include financial transactions such as: a check item debit in a bank statement, a credit debit in a VISA statement, or a shipping charge in a Federal Express bill, and non-financial transactions such as laboratory test results sent from a laboratory to a hospital.

Transaction Type—A unique type of transaction, e.g. a line-item in a customer bill.

Transmission Receipt—An electronic document which serves to notify a user as to whether a data transmission has been successfully or unsuccessfully transmitted.

Transmission Receipt Database—A database storing all Transmission Receipts received.

Transactions Table—The database file containing a list of processed transactions for all Subscribers.

User—A person who is operating a specified section of the system herein described, at a given time.

Work Directory—A temporary directory created by the PrepServer Software and used by the PreProcessor as work space in which to carry out data translation and manipulation processes.

Description of Exemplary Embodiments

In the foregoing definitions, it should be appreciated that the system herein described is composed of subsystems which perform their tasks simultaneously, or in a rapidly alternating, interactive fashion. Because a written specification requires putting these parallel processes into a linear explanation, discussions of such processes are presented here in a manner that requires moving back and forth between figures, system components, events, and stages within the processes.

In one embodiment, shown in the block diagram of FIG. 1, the system of the present invention extends over three or more sites—that of one or more Provider Sections 10, that of Main Processing Section 20, and that of one or more Subscriber Sections 30. These sites are linked, for example, by telecommunication methods such as modems (depicted as Telecommunications Links 11 and 21).

At Provider Section 10, Computer 52A may be one such as an IBM-compatible PC with a CPU such as Pentium Chip (Intel Inc., Santa Clara, Calif.), Main Memory such as Random Access Memory in quantities such as 8 Megabytes, and with peripheral devices such as a Monitor (NEC, Woodale, Ill.), a Mouse (Microsoft Corp., Redmond, Wash.), a Keyboard (Microsoft Corp., Redmond, Wash.), Mass Storage such as a hard drive (Maxtor, San Jose, Calif.) or other mass storage. Telecommunication Means such as Modem 54J, such as the Sportster 14.4 (U.S. Robotics, Skokie, Ill.) may be used. The modem requires the use of telecommunications software such as that provided RipTerm and RipPaint (Telegraphics, Huntington Beach, Calif.). The computer may use an operating system such as MS-DOS 6.0 (Microsoft Corp., Redmond, Wash.). A Printer such as one by Hewlett-Packard (Boise, Id.) may also be attached.

Hardware and software components in the telecommunication-linked system herein described can be relocated to various physical locations and still perform their respective functions, as will be shown in the discussion of FIGS. 2A–2C. With the exception of the FIGS. 2A–2C discussion, however, for the purposes of further discussion in this document, an embodiment of the Provider Section 10 will be considered to include Provider Section Telecommunications Software, a Data Provider File Management Program, Provider data Files, and hardware such as that previously described.

In one embodiment, at Main Processing Section 20, there are preferably two or more computers joined together in a network, using standard hardware such as an Ethernet card (Western Digital, Irvine, Calif.), and software such as Netware 3.1 by Novell Inc., (Provo Utah). A first computer, ComServer Most 76A, operates under the control of ComServer Software, described later, allowing it to specialize in communications via links 11 and 21, with other computers, such as Provider Section 10, PrepServer Most 78A, and Subscriber Section 30. A second computer, PrepServer Host

78A operates under the control of PrepServer Software, later described, which enables PrepServer Host 78A to specialize in data processing. ComServer Host 76A, and PrepServer Host 78A may each have the same hardware and operating system as the computer above described at Provider Section 10, except, of course, that ComServer Host 76A, and PrepServer Host 78A will also run the specialized software programs collectively referred to as ComServer Software and PrepServer Software. As an addition exception, only ComServer Host 76A requires a modem if both PrepServer Host 78A and ComServer Host 76A are used together. Multiple Modems 54A connected to the network via Multiplexer 74B such as the Wildcat! BBS IM, MultiPort multiplexer (Mustang Software, Bakersfield, Calif.) may be used in order to improve the efficiency of remote data reception and transmission from multiple parties simultaneously.

Hardware and software components in the telecommunication-linked system herein described can be relocated to various physical locations and still perform their respective functions as will be shown in the discussion of FIGS. 2A–2C. With the exception of the FIGS. 2A–2C discussion, however, for the purposes of further discussion in this document, an embodiment of the Main Processing Section 20 will be considered to include ComServer Software, PrepServer Software, PreProcessors, Billing and Administrative System, and hardware such as that previously described.

At Subscriber Section 30, the computer hardware and software may be exactly the same as that described for Provider Section 10. Additionally, the User of Subscriber Section 30 will have application software which requires input data files to be in a certain formats—hence giving the User of Subscriber Section 30 the need for using the system herein described. Examples of such Subscriber's Application Software include the accounting package called Platinum (Platinum Software, Irvine Calif.), spreadsheets such as Excel (Microsoft Corp., Redmond Wash.), and databases such as dbase (Borland, Scotts Valley, Calif.).

Hardware and software components in the telecommunication-linked system herein described can be relocated to various physical locations and still perform their respective functions as will be shown in the discussion of FIGS. 2A–2C. With the exception of the FIGS. 2A–2C discussion, however, for the purposes of further discussion in this document, an embodiment of the Subscriber Section 30 will be considered to include Subscriber Telecommunication Software, MidFormat Files, at least one PostProcessor, Subscriber's Application Software, and hardware such as that previously described.

A set of data that a Provider Section 10 sends to Main Processing Section 20 is called Provider Data File 115. The content of Provider Data File 115 may be accounting and financial information including but not limited to payroll transactions, bank transactions, bills, and cash receipts. The data may also be any of a wide variety of non-financial types, such as the values that a laboratory has found for specific attributes of a patient's blood sample, such as hemoglobin, hematocrit, sodium, potassium, and chloride levels. The format of these data can be divided into general categories, including, but not limited to ASCII, ANSI X.12, EDIFACT, Binary Files with Headers, graphics files, sound files, video files, and alternate compression method files. Such general categories may be further subdivided into an array of different formats. ASCII, or EBCDIC files, for example, may appear in several formats including, but not limited to Carriage return, Line Feed, Fixed Width without

Carriage Return, and delimited files. The particular format of a Provider Data File 115 is generically referred to herein as the Provider's Data Format. Regardless of the format used, a Destination Address 117, which describes who the data is for, accompanies the actual data within Provider Data File 115.

In a typical pattern of information flow, a Provider Data File originating on Computer 52A of Provider Section 10 is sent via Modem 54J and Telecommunications Link 11 to Main Processing Section 20 via an array of one or more Modems 54A through 54I and Multiplexers 74 coordinated by one or more ComServer Hosts 76A, 76B, 76C. After the data file is processed in an associated one of PrepServer Host 78A, 78B, 78C, the data, now in the form of a MidFormat File, is sent via one or more ComServer 76A, 76B, 76C, and the multiplexed array of one or more Modem 54A through 54I, through Telecommunication link 21 to Subscriber Section 30, where the data is further translated, if necessary, and utilized within a computer application. Box 40 represents the numerous other Provider Sections and Subscriber Sections with which the system may simultaneously or sequentially interact.

FIGS. 2A, 2B, and 2C illustrate alternative configurations for the physical location of the system's components, in which there are two rather than three sites through which data must pass. Specifically, Main Processing Section 20 does not have a separate physical location, and instead resides in Provider Section 10 only, or Subscriber Section 30 only. Main Processing Section 20 may also be parceled between Provider Section 10 and Subscriber Section 30. Furthermore, other components such as Subscriber Section 30 may be parceled. As long as the components are somewhere on the system, all necessary tasks can be accomplished in accordance with this invention. Thus, a given data file may be transmitted directly from Provider Section 10 to Subscriber Section 30, with the appropriate translation being provided.

In FIG. 2A, Provider Section 10 uses Main Processing Section software on its own computer data storage device such that it can send MidFormat Files to Subscriber Section 30, which can subsequently use PostProcessors to bring the data into the required format for loading it into the Subscriber's Application Software.

In FIG. 2B, Subscriber Section 30 has Main Processing Section 20 within it, in addition to its usual components. With the system configured in this manner, Subscriber Section 30 can receive files in their native Provider's Data Format, and perform all necessary processing, including translation, on site.

In FIG. 2C, Provider Section 10 has incorporated not only the components of Main Processing Section 20, but also PostProcessors. In this configuration, Provider Section 10 can send Destination Format files to Subscriber Section 30, ready to be used by the Subscriber's Application Software.

Of course, there are a wide variety of intermediate variations possible in which various components of the invention, such as PrepServer Software, PreProcessor, and PostProcessor, reside on the different machines through which information passes. It can be appreciated that cooperative behavior of the multiple computers interacting under the directions of the invention's software makes the physical location of any given component unimportant to the overall process.

When the system is implemented without a separate location for Main Processing Section 20, a separate Communications Server Host 76 computer may not be desirable, since a smaller volume of transmissions will likely pass

through Main Processing Section 20, increasing the feasibility of more simple telecommunications setups.

FIG. 3 shows an overview of one embodiment, demonstrating the main features of the system architecture at a process level, each block corresponding to one or more detailed Figures on successive sheets of the drawings. In Block 100 activities occurring at Provider Section 10 are highlighted, and is detailed in FIG. 4. Blocks 200, 300, and 400 highlight activities occurring at Main Processing Section 20, and are further detailed in FIGS. 5, 6A-C, and 7, respectively. Blocks 500 and 600 highlight activities occurring at subscriber Section 30, and are further detailed in FIGS. 8 and 9, respectively.

In Block 100, Provider Data Files 115 generated by Data Provider 113, and residing on Provider Section 10 are transmitted to Main Processing Section 20 via Telecommunications Link 11.

In Block 200, Provider Data Files 115 received via Telecommunication Link 11 and ComServer Software 201 is stored in MPS Incoming Data Box 355.

In Block 300, PrepServer Software 354 copies Provider Data File 115 from MPS incoming Data Box 355, to Work Area 366. Subsequently, the file is format-translated from the native format of Provider Data File 115, to that of a MidFormat File 342. This resultant MidFormat File 342 is then moved to MPS Outgoing Data Box 371.

In Block 400, ComServer Software 201 at Main Processing Section 20 forms Telecommunications Link 21 with the telecommunications software of Subscriber Section 30. MidFormat Files 342 that are in MPS Outgoing Data Box 371 which correspond to the particular Subscriber Section 30 with which contact is made, are transmitted over Telecommunications Link 21, to Subscriber Section 30.

In Block 500, Subscriber Section 30 receives MidFormat Files 342 from Main Processing Section 20, and stores these files.

In Block 600, PostProcessor 60 translates MidFormat Files 342 into Destination Format Files 610—the format needed by Subscriber Application Software 620. Destination Format Files are then inserted into Subscriber Application Software 620.

FIG. 4 shows in more detail one embodiment of events occurring within Block 200 of FIG. 3. Steps 110, 120, 130, 180, and 190 collectively form Data Provider File Management Program 105. Data Provider File Management Program 105 works in conjunction with Provider Section Telecommunication Software 142. Of course, these programs typically run concurrently with Main Processing Section telecommunications software while data is being downloaded—a process shown in detail in FIG. 5.

In step 110, Provider Data Files 115 are generated by Data Provider 113 by means well known in the art. One simple example of this is a Data Provider 113 manually entering data pertaining to a specific Data Subscriber into a database application such as dbase (Borland, Scotts Valley, Calif.). Adding a Destination Address 117 to such a data file is performed by a variety of means, including simply typing in an alphanumeric string into a designated field of the file, the string being uniquely representative of the specific Subscriber that the data is intended to ultimately reach. An example of this is a bank that enters the social security number of a client into a certain field of their file of cleared bank checks. In step 120, Provider Data File 115 with Destination Address 117 is loaded into Provider Section 10 computer, into Outgoing Data Box 128. Here, this particular process halts, and does not proceed to step 130, until

prompted by Provider Section Telecommunications Software 142. Provider Section Telecommunications Software 142 is initiated in step 140 by any of a variety of methods, including being set to initiate at a specific time, and initiating when a call is received from ComServer Software 201 of Main Processing Section 20. These methods are discussed in detail in the discussion of FIG. 5, steps 235 and 245. The basic problem that all of these methods address is that the quality of data ultimately delivered to Data Subscribers 33 must be ensured. Not having safeguards would be to risk obtaining an incorrect or error-ridden data set—either accidentally or by the intent of some third party posing as a provider. Similarly, a party other than the correct Subscriber Section 30 must not be allowed to receive proprietary and potentially sensitive data files belonging to another. Hence, caller validation procedures, coordinated, for example, by Main Processing Section 20, become important.

In one embodiment, the data transmission process is invoked in step 140, where the uploading process is initiated via Telecommunications Software 142. In step 150, Provider 113 connects with ComServer 152 and upon being validated by the methods discussed in the description of FIG. 5, receives clearance to transmit a copy of Provider Data File 115 to Main Processing Section 20. At this point Provider Data Files 115 that had been placed in Provider's Outgoing Data Box 128 in step 120 are read in step 170, copies being transmitted in step 160 over Telecommunications Link 11, to Main Processing Section 20. In step 180, Telecommunications Software 142 determines whether a copy of Provider Data File 115 has been transmitted successfully. Successful data transmission is signaled to Main Processing Section 20 by a response produced by the modem protocol. Most standard modem protocols including X, Y, and Z modem protocols have internal error checking means to carry out such a signaling process. If Provider Data File 115 has been successfully transmitted, then the Provider Data File 115 duplicate remaining in Outgoing Data Box 128 is "moved" to History Data Box 135 in step 130, for example by simply renaming the subdirectory. If Provider Data File 115 is not successfully transmitted, then an error appears in Provider Error Log 187, in step 185. Data Provider File Management Program 105 is thus completed.

Using a modem, Provider Section 10 can either directly transmit Provider Data Files 115 to Main Processing Section 20, or electronically mail this information. Of course, there are many other means of electronic data transmission known in the art that are also suitable for this purpose, within the framework of the present invention. Any of a variety of modem protocols may be used, such as X, Y, or Z modem protocols.

Alternative methods of communication and electronic mail between separately located sections of this system include the use of on-line services such as CompuServe, America Online, Prodigy, or infrastructures such as the Internet, or the Information Superhighway, each with their respective electronic mail and file transfer capabilities.

In another alternative embodiment, non-modem transmission means such as ISDN, and Internet IP/FTP transmit the data from Provider Section 10 to Main Processing Section 20.

The information transmitted by Provider Section 10 can be encrypted with a combination of security techniques, including, but not limited to data encryption and security measures. Methods of data encryption are well known in the art. Examples of data encryption methods are RSA Data Security (Redwood City, Calif.) software solutions, and the use of the proposed "Clipper" chip hardware solution.

Data Provider Telecommunications Software 142 used to support the telecommunication links necessary to transfer Midformat Files 342 from Provider Section 10 to Main Processing Section 20 may be one such as RipTerm (TeleGrafix Communications Inc., Huntington Beach, Calif.). A simple user interface that gives the necessary instructions to the telecommunications software to carry out the previously tasks may be assembled by well-known methods using software such as RipPaint (TeleGrafix Communications Inc., Huntington Beach, Calif.).

FIG. 5 details an embodiment of Block 200 of FIG. 3, in which ComServer Software 201 at Main Processing Section 20 receives Provider Data files 115, and stores them in a specific MPS Incoming Data Box 355.

In step 205, ComServer Software 201 is started by the user of Main Processing Section 20.

In step 210, ComServer Software 201 continuously scans modems 54 and multiplexers 74 for a Provider 12. In an alternative embodiment, ComServer Software 201 scans modems 54 and multiplexers 74 for a Provider 12 at predetermined time intervals. In another alternative embodiment, ComServer Software 201 scans modems 54 and multiplexers 74 for a Provider 12 at the command of a Main Processing Section 20 user.

In Step 215, ComServer Software 201 determines whether a Provider Section 10 is calling Main Processing Section 20, requesting a connection.

In Step 220, ComServer Software 201 determines that a call is not made, or that program termination has been otherwise requested.

In Step 225, ComServer Software 201 is terminated, as it has been instructed by the program's flow. In an alternative embodiment, ComServer Software 201 is terminated on demand by Main Processing Section 20 user.

In Step 230, ComServer Software 201 activates the Billing and Administrative System 373 database. This database stores information regarding all valid Providers 12, their respective Provider Sections 10, and corresponding passwords and security levels.

In Step 235, ComServer Software 201 accesses Billing and Administrative System 373 database to lookup and determine whether the calling party is a valid Provider Section 10.

Step 240 shows an example of a calling party dialing into Main Processing Section 20.

In Step 245, ComServer Software 201 determines whether to allow the calling party to connect to Main Processing Section 20 based upon the security password and security level.

In Step 250, ComServer Software 201 disconnects from invalid calling parties. In one embodiment, ComServer Software 201 records the calling party phone number along with a date-stamp and time-stamp in the Billing and Administrative System 373 database. Finally, ComServer Software 201 prepares to scan and detect the next calling party.

In Step 255, ComServer Software 201 connects to valid Provider 12 and grants Subscriber Section 30 access to the Main Processing Section 20. ComServer Software 201 then responds to Provider 12 request to receive Provider Data File 115.

In step 260, ComServer Software 201 determines which Provider 12 MPS Incoming Data Box 355 should be used and begins storing Provider Data File 115.

In Step 270, ComServer Software 201 sends a Data Transmission Receipt 202 to Provider Section 10. This

receipts tells Provider 12 whether Provider Data File 115 transmitted successfully. Finally, ComServer 201 disconnects from Provider 12 and prepares for the next caller.

ComServer Software 201 is used to support the telecommunication links necessary to receive MidFormat Files 342 from Provider Section 10 to Main Processing Section 20 may be one such as RipTerm (TeleGrafix Communications Inc., Huntington Beach, Calif.). A simple user interface that gives the necessary instructions to the telecommunications software to carry out the previously tasks may be assembled by well-known methods using software such as RipPaint (TeleGrafix Communications Inc., Huntington Beach, Calif.).

Validation, as shown in Step 235 is performed in any of a variety of ways. In one embodiment, Main Processing Section 20 employs the caller identification method to determine whether the caller is valid. By this caller ID method, the telephone number of any party attempting to establish connection with Main Processing Section 20 is screened by Main Processing Section 20 by matching the telephone number of the caller to a list of valid Provider numbers. This valid number list is maintained by a database internal to Main Processing Section 20, to ensure that the calling party is a valid Provider Section 10. If the telephone number is not that of a valid Provider Section 10, Main Processing Section 20 does not establish a connection, and Main Processing Section 20 time-stamps, date-stamps, and records that invalid phone number. If the number captured, on the other hand, represents a valid Provider Section 10, Main Processing Section 20 allows the transfer of Provider Data files 115 from Provider 10 to Main Processing Section 20.

In an alternative embodiment, Main Processing Section 115 prompts Provider Section 10 for a valid log-in name and/or a password. The password is preferably changed frequently. The entry of a log-in name and/or password may be automatic or manual on the part of Provider Section 10.

In an alternative embodiment, data transmission is initiated by a call from Main Processing Section 20 to Provider Section 10, rather than vice-versa.

In an alternative embodiment, a connection between Provider Section 10 and Main Processing Section 20 is established upon securing access through existing security systems such as a Firewall.

In an alternative embodiment, the "call-back" method is used. In this case, Provider Section 10 establishes a connection with Main Processing Section 20. Main Processing Section 20 verifies the validity of Provider Section 10 by means such as those described above. The communication link is then disconnected. Next, Main Processing Section 20 calls back to, and connects with Provider Section 10, and proceeds to download Provider Data Files 115. The need for a call to be returned by the prospective data recipient (in this case, Main Processing Section 20) provides an additional measure of security to the system.

In alternative embodiments, any combination of the above mentioned validation/security methods are used. For example, the "caller ID" method may be used to determine the origin of a call. If the calling number is deemed valid, then Provider Section 10 is prompted by Main Processing Section 20 to provide a log-in name and password.

If the transmission is successful, Main Processing Section 20 puts the data into a specified MPS Incoming Data Box 355 and a successful data transmission receipt is transmitted to Provider Section 10. Main Processing Section 20 simultaneously creates and records a successful data transmission

receipt, which is stored in a database within Main Processing Section 20. Each successful data transmission receipt has a unique transmission tracking number assigned to it. This tracking number is generated by Main Processing Section 20 for audit purposes. In addition to the unique transmission tracking number, each successful data transmission receipt contains a date and time stamp, the file name, and the file size. The cumulative set of successful data transmission receipts enable those individuals who service Main Processing Section 20 to monitor the performance of the system, and to serve as a medium for historical/legal purposes.

In an alternative embodiment, Main Processing Section 20 generates a successful data transmission receipt created at Provider Section 10 location, and which is then sent to Main Processing Section 20 and to Data Provider History Box 135 by default, if no command to the contrary is returned from Main Processing Section 20.

Once data 118 is received by Provider Section 10, Main Processing Section 20 stores and archives the data provided by Provider Section 10 in its original, Data Provider's Format. Storage media employed by Main Processing Section 20 may include magnetic and non-magnetic storage devices including, but not limited to hard disks, tapes, magneto-optical disks, compact disks and memory chips. Of course, other forms of data storage are known in the art, and it will be appreciated by persons of ordinary skill in the art in light of the teachings of this invention, such other forms of data storage may be readily used.

FIG. 6A describes one embodiment for carrying out the processes that occur at Main Processing Section 20. PrepServer Software 354 is the specialized operating system of Main Processing Section 20 that performs functions including, but not limited to data capture, data translation, data forwarding, Subscriber Section 30 billing, error tracking, and data and system security. Additionally, PrepServer Software 354 works along side operating systems such as DOS (Microsoft Corporation, Redmond, Wash.), Netware (Novell, Inc. Provo, Utah), and UNIX to perform functions herein described.

The PrepServer Software 354 begins in step 352. Once initiated, PrepServer Software 354 is instanced and assigned, in step 353, a set of MPS Incoming Data Boxes 355 to patrol.

In one embodiment, PrepServer Software 354 continuously scans MPS Incoming Data Boxes 355 of each Provider Section 10, searching for new data files in step 357.

In an alternative embodiment, PrepServer Software 354 scans MPS Incoming Data Boxes 355 of each Provider Section 10 at predefined times.

In another alternative embodiment, PrepServer Software 354 scans all of MPS Incoming Data Boxes 355, but the scanning is manually initiated.

In yet another alternative embodiment, PrepServer Software 354 is dormant until it is triggered by a signal announcing that new data has been successfully received.

In step 356, PrepServer Software 354 scans its assigned MPS Incoming Data Boxes 355, looking for Provider Data Files 115 to be processed. Step 357 defines the collection of MPS Incoming Data Boxes 355 that PrepServer Software 354 is assigned to patrol. Step 358 discerns whether a Provider Data File 115 was found and is available. If no Provider Data File 115 is available, then step 359 is invoked, in which PrepServer Software 354 scanning is completed. If a Provider Data File 115 was found, and is available, step 360 is invoked. In this step, PrepServer Software 354 discerns the identity of Data Provider 113 and determines

15

which PreProcessor 363 is needed, based upon which MPS Incoming Data Box contains a Provider Data File 115, or by Destination Address 117 residing inside of Provider Data File 115.

PreProcessor Directory 363 warehouses the various specialized PreProcessors, thus functioning as a type of database that holds actual programs rather than just representational data. Once it has been determined which particular PreProcessor 363 is needed to translate from the particular Provider Data Format of the Provider Data File 115 found, PreProcessor Directory 362 is accessed to retrieve the appropriate Pre-Processor 363 in step 361. Next, Work Area 365 is blanked or cleared in preparation for use by PreProcessor 363 in step 364. Step 366 represents data being moved to and from Work Area 365 where data processing occurs. Work Area 365 is a temporary directory that is dedicated to the need for PrepServer Software 354 to have an area of information storage in which data can be manipulated.

In an alternative embodiment, PreProcessor 363 is capable of translating similar Midformat files 342. Such a PreProcessor 363 derives its capabilities by accessing a resource database which describes how to translate the given MidFormat Files 342. The resource database describes information in MidFormat Files 342 such as the included fields, the size of a given field, and the type of data. This results in one PreProcessor 363 being used to translate any similar MidFormat Files 342 from any provider. In the best case, there would be one PreProcessor 363 to translate all Midformat files 368. In addition to PreProcessor 363 and support libraries, a graphical editor is used to create and modify the resource database used by such a PreProcessor 363. This allows PreProcessor 363 systems to be quickly set up to add and translate new Midformat files 342.

Because of the scalability of Main Processing Section 20, multiple PrepServer Software 354 programs, as well as multiple PrepServer Hosts 78, may be operated simultaneously.

Data processing involves PreProcessor 363 translating Provider Data Files 115 to a MidFormat, storing the resultant MidFormat Files 342, counting the number of transactions in MidFormat, and finally returning Success Flag 369, or a Failure Flag 370 to PrepServer Software 354, indicating whether or not PreProcessor 363 successfully translated Provider Data File to which it had been assigned. These last events occur within step 367 of FIG. 6A. Further details of one embodiment of PreProcessor 363 function are shown in FIG. 6B.

FIGS. 6B and 6C describe a method and data structures used in an embodiment for creating MidFormat File 342 from a Provider Data File 115.

In step 301, PreProcessor 363 is selected and sent into operation by PrepServer Software 354.

In step 305, PreProcessor 363 gets Subscriber ID 308 located in Provider Data File 115 header. This is done by looking in a given byte location for a sequence of bytes making up Subscriber ID 302, or by finding an attribute-value pair and returning the value found.

In step 310, using Subscriber ID 302, PreProcessor 363 gets Subscriber Job Number 311 from Subscriber Table 314. In one embodiment, PreProcessor 363 accesses Subscriber Table 314, a database table or file such as one created in Paradox (Borland, Scotts Valley, Calif.) to get Subscriber Record 307, then returns Subscriber Job Number 311 from that record.

In step 315, using Subscriber ID 308 and Subscriber Job Number 311, a unique filename is generated for MidFormat

16

File 342. In one embodiment, a unique filename is generated by using a three letter Subscriber ID as a prefix, and appending a five letter job number after first incrementing it. For example, if Subscriber ID is "AAA" and the job number is "00000", then the unique filename produced would be "AAA00001". Note the job number is incremented and saved for the next time a unique filename must be generated. Also note the job number is using a base-36 notation, whose values belong in the set [0-9A-Z]. This means the number of unique filenames which can be generated for this one Subscriber Section 30 is 36 to the 5th power, i.e. approximately 60.4 million filenames. Of course, a wide variety of other file naming schemes, such as the many known in the art, may also be used.

In step 320, PreProcessor 363 creates an empty MidFormat File 342 in Work Area 365, in order to serve as a matrix in which translated data is placed.

In step 325, PreProcessor 363 obtains Subscriber Translation Information 312 from Subscriber Table 314. Subscriber Translation Information 312 describes the format of Provider Data File Transactions 302, which collectively constitute most of Provider Data File 115, and which PreProcessor 363 translates to MidFormat File Transactions 304. Subscriber Translation Information 312 includes the names, types and sizes of the fields it will be accessing in Provider Data File Transaction 302, and serves as an interpretive guide by which PreProcessor 363 can completely read Provider Data File 115. Attribute-value pairs may also be used to describe the fields found in Provider Data File Transaction 302.

In step 330, PreProcessor 363 attempts to get a Provider Data File Transaction 302 from Provider Data File 115. If it finds one, it continues to step 335 to translate the found transaction, in which PreProcessor 363 manipulates Provider Data File 115 according to the rules specified in PreProcessor 363. Such rules may include, but are not limited to, instructions for the transposition of specific fields, and other known means for accomplishing file format translation. If a Provider Data File Transaction 302 is not found, PreProcessor 363 assumes the end of Provider Data File 115 is reached, and proceeds to step 350.

In step 335, using Subscriber Translation Information 312 found on Subscriber Table 308, PreProcessor 363 takes a Provider Data File Transaction 302 and generates the equivalent MidFormat File Transaction 304. In step 340, the generated MidFormat File Transaction 304 is written to MidFormat File 303.

Once PreProcessor 363 successfully manipulates the data file, the resulting data file is now called MidFormat File 342. The data file is called "MidFormat" because it is in a standard format from which it can be ultimately converted into any destination format.

In step 345, a counter keeping track of the number of transactions processed so far is incremented. After this is done, PreProcessor 363 returns to step 330 and attempts to obtain the next Provider Data File Transaction 302. In step 350, PreProcessor 363 cannot find any more Provider Data File Transactions 302 to process, so it records the number of transactions it processed in Transactions Table 313, and returns to PrepServer Software 354 in step 352.

Returning to FIG. 6A, in step 371, if PreProcessor 363 returns Success Flag 369 in step 371, then step 370 is invoked. In step 370 PrepServer Software 354 distributes MidFormat File 342 from Work Area 363 to the appropriate MPS Outgoing Data Boxes 371, using Subscriber Destination Address 117 as a guide 371A, 371B and 371C are

examples of different Subscriber Outgoing Data Boxes 371, each dedicated to a specific file type and Subscriber 33. After the distribution occurs, PrepServer Software 354 updates Billing and Administrative System 373 with Subscriber Transaction Count 374 in step 372. Billing and Administrative System 373 is a database containing a listing of Subscribers and relevant account information that allows subscribers to be appropriately billed for their usage of the herein described system. This usage count is continually updated in step 375. In Step 376, PrepServer Software 354 moves Provider Data file 115 from MPS Incoming Data Box 355 to MPS History Box 377. PrepServer Software 354 has then completed its process.

If PreProcessor 363 returns Failure Flag 370, however, step 378 is invoked, in which PrepServer Software 354 updates Error Log 379 with a report of PreProcessor 363 Failure flag 370. In step 380, PrepServer Software 354 moves Provider Data File 115 from MPS Incoming Data Box 355 to MPS Failed Data Box 373. PrepServer Software 354 has then completed its processing.

In step 368, PreProcessor 363 sends a response to PrepServer Software 354 indicating that either a successful translation has occurred or has not occurred. In one embodiment, the final task of PreProcessor 363 is to submit a response, such as the answer "1" signifying a successful translation or such as the answer "0" signifying an unsuccessful translation.

In an alternative embodiment, the computer operating system, such as DOS, signifies the success or failure of the translation. Upon successful completion of PreProcessor, operating systems such as DOS return a default response of the answer "1". This response is then interpreted by PrepServer Software 354 to perform subsequent events. Upon a failure to complete execution of PreProcessor 363, operating systems such as DOS return a default response of the answer "0". This response is then interrupted by PrepServer Software 354 to provide error messages to the administrator of PrepServer Software 354.

In the event that a data file 115 fails to move from MPS Incoming Data Box 355, or fails to move from Work Area 365 to MPS Outgoing Data Box 371 the data file is moved to the Failed Data Box. The Failed Data Box is an area which triggers a user error report and screen display specifying that a movement of data has occurred and that examination is necessary to rectify the error.

Numerous failures can occur while data processing. These events include, but are not limited to failures in hardware, software, power supply, and of telecommunication lines. In the event that a data translation fails, the data file is moved from Work Area 365 to a Failed Data Box subdirectory. The Failed Data Box subdirectory is an area which triggers a user error report and/or screen display specifying that a translation error has occurred and that examination is necessary to rectify the error. The failure is triggered by PreProcessor 363 or by the operating system as previously discussed.

In step 370 PreProcessor 363 moves MidFormat File 342 from Work Area 365 to the correct MPS Outgoing Data Box 371. PrepServer Software 354 selects which MPS Outgoing Data Box 371 is appropriate in the manner previously discussed.

Additionally, PrepServer Software 354 creates a unique transaction assigned to each specific MidFormat File 342. This transaction contains a unique transaction tracking number. This tracking number is generated by PrepServer Software 354 for audit purposes. Additionally, this unique transaction contains a date-stamp, time stamp, the file name, and

the file size. The cumulative set of successful data transmission receipts enables the user of Main Processing Section 20 to monitor the performance of the system, and to serve as a medium for historical/legal purposes.

The system utilizes numerous internal databases. These internal databases store historical and referential data about security, data transmission, PrepServer Software 354, PreProcessors 363, translations, Providers Sections 20, Subscriber Sections 30, billing information, data tests later to be described, and internal system logs.

Provider Sections 20 and Subscribers Sections 30 are collectively referred to as "customers", and the database that contains information about them is called the Customer Definition Table (see Table 1). The purpose of this database is to store relevant information about the customers such as the customer's name, customer's identification, address, etc.

The database of Incoming Data Boxes 355 and Outgoing Data Boxes 371, is referred to as the Mail Box Database. The purpose of this database is to store relevant information about which customers own the mail boxes, where the mail boxes reside, which PreProcessor 363 performs work on the data contained in the mail box, etc., as shown, for example, in Table 2. From the moment that a data file arrives from Provider Section 10, the data file's ultimate destination is predetermined—it will go to a single specific Subscriber Section 30.

In one embodiment, PrepServer Software 354 recalls from the internal database containing the successful transmission data receipts which MPS Incoming Data Box 355 the data file originally arrived in. Each Incoming Data Box 355 is assigned to only one Subscriber Section 30, to only one transaction type, and to only one MPS Outgoing Data Box 371 with a specific transaction type. Hence, a Subscriber Section 30 may have many MPS Outgoing Data Boxes 371 assigned to it and each Outgoing Data Box 371 representing a single specific type of transaction. For example, Subscriber X might have one MPS Outgoing Data Box 371 assigned to him for holding his cleared bank check files, and a MPS Outgoing Data Box 371 for holding his payroll data files. Because both boxes are reference to the same party, however, they will be correctly transmitted to the same, appropriate Subscriber Section 30.

In an alternative embodiment, PrepServer Software 354 detects the appropriate destination from MidFormat File 342 characteristics other than a Subscriber Destination Address 117. Such characteristics include but are not limited to the file name, the file size, the file header, and the file trailer. By such cues, PrepServer Software 354, in one embodiment, determines the target MPS Outgoing Data Box 371 and moves MidFormat File 342 to that specific box. In an alternative embodiment, PrepServer Software 354 detects from the Provider's 12 log-in name and or log-in password the MPS Outgoing Data Box 371 to select. Based upon the log-in name and or log-in password, PrepServer Software 354 then selects the corresponding MPS Outgoing Data Box 371 based upon a database of log-in names and/or log-in passwords with corresponding MPS Outgoing Data Boxes 371, and moves the MidFormat File 342 to the target box. In an alternative embodiment, Provider Section 10 specifies which Outgoing Data Box 371 to use, and PrepServer Software 354 records this information. Based upon this data box identity, PrepServer Software 354 looks up the target data box and puts MidFormat File 342 into the appropriate MPS Outgoing Data Box 371. In yet an additional alternative embodiment, a combination of these alternatives is used to determine the MPS Outgoing Data Box 371.

Table 3 shows some aspects of a database entitled Usage-Transactions Table for Tracking and Accounting. This database contains records of transactions, each transaction representing a translation to MidFormat that has been performed, when it occurred, when it was transmitted, and other such pertinent information.

Another internal database holds Data Transaction receipts for each customer. Specifically, each transaction contains information such as the number of transactions that have been translated, the date and time processed, and the identity of the Subscriber. The transactions in this database form the basis for customer billing purposes as well as for audit and legal purposes.

A database called the Sys_log, such as the example shown in Table 4, defines the database of transactions for each warning and error that occurs in Main Processing Section 20. For example, all telecommunication links, whether coming in or out, are recorded and date stamped, as are attempts to access the ComServer Host 76A and ComServer Software 201 without valid phone numbers, log-in names, and passwords.

Another database called Batchnum, such as the example shown in Table 5, defines the transaction tracking number assigned to each data transmission receipt. The purpose of this table is store, assign, and increment a unique number each time a data transmission receipt is sent to a Provider Section 10 and Subscriber Section 30.

Another database contains all data transmission receipts. The purpose of this table is to keep a record of all receipts for audit and legal purposes. Each transaction is date and time stamped, has a unique batch number assigned to it, the file size transmitted, and a description of whether the transfer was successful or not.

Referring again to FIG. 7, in step 405, ComServer Software 201 is started by the user of Main Processing Section 20.

In step 410, ComServer Software 201 continuously scans modems 54 and multiplexers 74 for a Subscriber Section 30. In an alternative embodiment, ComServer Software 201 scans modems 54 and multiplexers 74 for a Subscriber Section 30 at predetermined time intervals. In an alternative embodiment, ComServer Software 201 scans modems 54 and multiplexers 74 for a Subscriber Section 30 at the command of the Main Processing Section 20 user.

In Step 415, ComServer Software 201 determines whether a Subscriber Section 30 is calling Main Processing Section 20.

In Step 420, ComServer Software 201 determines a call is not made.

In Step 425, ComServer Software 201 is terminated at predefined times.

In an alternative embodiment, ComServer Software 201 is terminated on demand by MPS 20 user.

In Step 430, ComServer Software 201 activates the Billing and Administrative System 373 database. This database stores the identity and information about all valid Subscriber Sections 30, their corresponding passwords and security levels, and their use of the system described in this patent application.

In Step 435, ComServer Software 201 accesses Billing and Administrative System 373 database to lookup and determine whether the calling party is a valid Subscriber Section 30.

Step 440 shows an example of a calling party dialing into the Main Processing Section 20.

In Step 445, ComServer Software 201 determines whether to allow the calling party to connect to Main Processing Section 20 based upon the security password and security level.

In Step 450, ComServer Software 201 disconnects from invalid calling parties. ComServer Software 201 records the calling party phone number along with a date-stamp and time-stamp in the Billing and Administrative System 373 database. Finally, ComServer Software 201 prepares to scan and detect the next calling party.

In Step 455, ComServer Software 201 connects to valid Subscriber Section 30 and grants Subscriber Section 30 access to the Main Processing Section 20. ComServer Software 201 then responds to Subscriber Section 30 request to transmit the MidFormat File 368.

In step 460, ComServer Software 201 determines which Subscriber Section 30 that MPS Outgoing Data Box 371 should be used and begins transmitting Midformat File 368 in Outgoing Data Box 371.

In Step 470, ComServer Software 201 sends a Data Transmission Receipt 402 to Subscriber Section 30. This receipt tells Subscriber Section 30 whether Midformat File 368 transmitted successfully. Finally, ComServer 201 disconnects from Subscriber Section 30 and prepares for the next caller.

This section describes how Main Processing Section 20 notifies Subscriber Section 30 and Main Processing Section 20 database that a successful data transmission has occurred. Prior to initiating contact with Subscriber Section 30, Main Processing Section 20 places data into Outgoing Data Box 371. A successful data transmission is signaled to Main Processing Section 20 by a response produced by the modem protocol. Most standard modem protocols, including X, Y, and Z modem protocols, have internal error checking means that will carry out this task for Main Processing Section 20.

Upon successfully transmitting MidFormat File 342 and the completion of the data transmission protocol, Main Processing Section 20 moves MidFormat File 342 from MPS Outgoing Data Box 371 to a MPS History Box.

As PrepServer Software 354 moves MidFormat File 342 from MPS Outgoing Data Box 371 to MPS History Box, a successful data transmission receipt is transmitted to Subscriber Section 30. Main Processing Section 20 simultaneously creates and records a successful data transmission receipt stored in Main Processing Section 20 database. Each Successful data transmission receipt has a unique transmission tracking number assigned to it. This tracking number is generated by Main Processing Section 20 for audit purposes. In addition to the unique transmission tracking number, each successful data transmission receipt contains a date and time stamp, the file name, and the file size. The cumulative set of successful data transmission receipts enable the user of Main Processing Section 20 to monitor the performance of the system, and to serve as a medium for historical/legal purposes.

In an alternative embodiment, Main Processing Section 20 generates a successful data transmission receipt which is sent to Main Processing Section 20, and to a Subscriber Section 30 History Box by default, if no command to the contrary is returned from Main Processing Section 20.

FIG. 8 shows a detail of one embodiment of the processes occurring within Block 500 of FIG. 3, primarily, Subscriber Section 30 receiving and storing MidFormat Files 368 from Main Processing Section 20. In one embodiment, Subscriber Section 30 initiates contact with Main Processing Section 20

with telecommunications link 21, for example by modem. Subsequently, Subscriber Section 30 is identified by Main Processing Section 20, and assessed as to whether the link is with a valid subscriber. Unless the call is deemed to be valid, the downloading of MidFormat Files 342 from Main Processing Section 20 does not begin.

In one embodiment, Main Processing Section 20 employs the caller identification method. By this caller ID method, the telephone number of any party attempting to establish connection with Main Processing Section 20 will be obtained by methods and devices well known in the art. The number obtained is screened by Main Processing Section 20 by matching the telephone number of the calling party to a list of valid Subscriber Section 30 telephone numbers. This valid number list is maintained by a database internal to Main Processing Section 20 to ensure that the calling party is a valid Subscriber Section 30. If the number is not that of a valid Subscriber Section 30, Main Processing Section 20 does not establish a connection, and Main Processing Section 20 time-stamps, date-stamps, and records that invalid phone number. If the number captured, on the other hand, represents a valid Subscriber Section 30, Main Processing Section 20 allows the transfer of the appropriate MidFormat Files 342 to Subscriber Section 30.

In an alternative embodiment, Main Processing Section 20 prompts Subscriber Section 30 for a valid log-in name and/or password. The entry of a log-in name and/or password may be automatic or manual on the part of the user of Subscriber Section 30.

In an alternative embodiment, data transmission is initiated by the call of Main Processing Section 20 to Subscriber Section 30, rather than vice-versa.

In another alternative embodiment, a connection between Subscriber Section 30 and Main Processing Section 20 will be established upon securing access through existing security systems such as a Firewall.

In an alternative embodiment, the "call-back" method is used. In this case, Subscriber Section 30 establishes a connection with Main Processing Section 20. Main Processing Section 20 verifies the validity of Subscriber Section 30 by means such as those described above. Telecommunication Link 21 is then disconnected. Next, Main Processing Section 20 calls back to, and connects with Subscriber Section 30, and then proceeds to transfer the appropriate MidFormat Files 342 to Subscriber Section 30. The need for a call to be returned by the prospective data source (in this case, Main Processing Section 20), provides an additional measure of security to the system.

In alternative embodiments, any combination of the above mentioned validation/security methods are used. For example, the "caller ID" method may be used to determine the origin of a call. If the calling number is deemed valid, then Subscriber Section 30 is prompted by Main Processing Section 20 to provide a log-in name and password.

Using a modem, for example, Subscriber Section 30 can either directly download MidFormat File 342 from Main Processing Section 20, or receive this information using electronic mail. Any of a variety of modem protocols may be used, such as X, Y, or Z modem protocols. Of course, there are other non-modem means of electronic data transmission known in the art that are also suitable for this purpose. A single modem can be used to transmit data to Subscriber Sections 30. Alternatively, a multiplexer can be used to handle several simultaneous transmissions to multiple Subscriber Sections 30. A multiplexer is a device well known in the art, which networks several modems and telephone lines

together. Such devices are commonly available. Examples include the "Wildcat BBS IM, MultiPort Multiplexer (Mustang Software, Bakersfield, Calif.)

The software used to support the Telecommunication Links 21 necessary to transfer Midformat Files 342 from Main Processing Section 20 to Subscriber Section 30 may be one such as RipTerm (TeleGrafix Communications Inc., Huntington Beach, Calif.). A simple user interface that gives the necessary instructions to the telecommunications software to carry out the previously described tasks may be assembled by well known methods using software such as RipPaint (TeleGrafix Communications Inc., Huntington Beach, Calif.).

MidFormat File 342 may be received by Subscriber Section 30 using a combination of security techniques, including, but not limited to data encryption and security measures internally employed by Main Processing Section, to be later described. Methods of data encryption are well known in the art. Examples of data encryption methods include RSA Data Security's (Redwood City, Calif.) software solutions, and the use of the proposed "Clipper" chip hardware solution.

Alternative methods of communication between separately located sections of this system include the use of on-line services such as CompuServe, America Online, Prodigy, or infrastructures such as the Internet, each with their respective electronic mail and file transfer capabilities.

In another alternative embodiment, non-modem transmission means such as ISDN, and Internet IP/FTP transmit the data from Main Processing Section 20 to Subscriber Section 30.

FIG. 9 represents one embodiment of the functions performed by PostProcessor 601 at Subscriber Section 30. PostProcessor 601 functions include recording the data transmission receipt, detecting MidFormat File 342, reading MidFormat File 342, translating the data in the file, validating the data in the file, appending information onto the data, inserting the data into the Subscriber's Application Software 620, and generating warning and errors displayed via the screen and paper reports.

In one embodiment, the user manually starts the telecommunication download process from within Subscriber's Application Software 620 in step 540. Once Midformat File 342 is successfully downloaded, the Telecommunication Software 542 automatically disconnects, and automatically initiates PostProcessor 601.

In an alternative embodiment, the user manually starts the telecommunication download process in Subscriber's Application Software 620. Once MidFormat Files 342 are successfully downloaded, the user of Subscriber Section 30 quits telecommunication package and manually starts PostProcessor 601.

In yet an additional embodiment, the user manually starts the telecommunication download process as an independent process outside of Subscriber's Application Software 620. An example of this method is using America Online's telecommunications software and interface to download a file previously uploaded to America Online by Main Processing Section 20. Once MidFormat File 342 is successfully downloaded, the user quits the Telecommunication Software 542 (in this example, America Online's package) and manually starts PostProcessor 601 to perform the tasks previously described.

Step 610 shows PostProcessor 601 recording Data Transmission Receipt 603. This receipt serves to provide the user of Subscriber Section 30 with acknowledgment that Mid-

Format File 342 has been either successfully or unsuccessfully downloaded and to let the user of Subscriber Section 30 know that the telecommunication download processing phase is complete. Additionally, Data Transmission Receipt 603 is stored in Transmission Receipt Database 607 to enable the user of subscriber Section 30 to subsequently review previous downloads for auditing and accounting purposes.

In step 620, PostProcessor 601 searches for MidFormat File 342. PostProcessor 601 detects the correct MidFormat File 342 by searching for a specific file name and/or looking up the specific file name in a specific data directory.

In step 625, PostProcessor 601 begins performing several tasks, starting with reading MidFormat File 342. In step 630, PostProcessor 601 subsequently records the data transmission receipt, displaying to the user of Subscriber Section 30 that a successful or unsuccessful data file exists. This process also involves discerning and displaying to the user of Subscriber Section 30 whether the MidFormat File 342 is the correct file, and discerning whether the data inside the file is complete and correct by using tests such as check sums, algorithms and other such tests, as is known in the art.

If any single one or a combination of these test results indicate that there is a problem with the data set, PostProcessor 601 halts all tasks. PostProcessor 601 then displays and/or print out a Warning/Error Report. Additionally, PostProcessor 601 suggests appropriate solutions to the problems listed in the Warning/Error Report. Each problem that can occur in PostProcessor is given a unique warning and error number and these warning and error numbers are stored in a PostProcessor 601 database along with the related suggested remedy(s). PostProcessor 601 provides remedy suggestions by looking up the warning/error number associated with each warning or error in the database, and displaying or printing the corresponding interpretations and suggested remedies.

In step 630, PostProcessor 601 reads MidFormat File 342 into memory where it manipulates MidFormat File 342 data in order to perform a format translation. The data manipulation is performed by a set of instructions provided by PostProcessor 601 source code. Each unique PostProcessor 601 has a specific set of manipulation instructions which are suited to the type of MidFormat File 342 being translated—for example whether it represents cleared bank checks, or payroll transactions—and the Destination Format required by Subscriber's Application Software 620. Such instructions may include format translation methods such as the rearrangement of data fields, the substitution of data fields using look up tables, the re-representation of numeric formats, the creation of data fields based upon PostProcessor 601 tests, the creation of data fields based specified by the user, the creation of data fields based upon lookup tables, the creation of data fields based upon Subscriber's Application Software 620, and other widely known data manipulation functions known in the art.

Additionally, PostProcessor 601 places the transactions in the format required by Subscriber's Application Software 620. This format, in any specific case, is generically referred to herein as Destination Format. These databases and database formats include but are not limited to BTrieve, Oracle, Sybase, Informix, Platinum, Great Plains, Solomon, Macola, Intuit, and other flat file and relational databases.

In step 635, one or more tests are performed to validate the data. One validation test is to compare the total transactions, dollar amounts, and other specified fields to control totals either in MidFormat File 342 or that are derived by PostProcessor 601.

A second test is to validate each field in a given transaction against data held within Subscriber's Application Software 620. This audit test helps to ensure that transactions and/or data fields belong to the particular Subscriber Section 30.

Another test involves using calculations such as algorithms. These tests involve specific calculations, such as $A+B/C$, $A^2+B^2=C^2$, $2A^2+4B^2$ the square root of $3C=X$, etc., that compare a data field or combination of data fields in the transaction against specified data in PostProcessor 601, or against specified data provided by the user of Subscriber Section 30, or against specified data from within Subscriber's Application Software 620. Another test involves Boolean logic routines, such as "less than", "greater than", "greater than or equal to". These routines compare a field or combination of fields in the transaction against specified data in PostProcessor 601, against specified data generated by PostProcessor 601, against specified data provided by the user of Subscriber Section 30, or against specified data provided by Subscriber's Application Software 620.

Another test involves statistical data. PostProcessor 601 generates statistics based upon a single or a combination of fields in MidFormat File Transactions 337. PostProcessor 601 then compares these statistics against specified data in PostProcessor 601, or against other specified data generated by PostProcessor 601, or against specified data provided by the user of Subscriber Section 30, or against specified data provided by the Subscriber's Application Software 620.

Of course, any combination of the above-described Boolean logic, statistical, and/or calculation/algorithm-based methods may be employed. In the event that a warning or an error occurs from one or a combination of these tests, the warning(s) or errors(s) are displayed and or printed to the user subsequent to the insertion of the transactions that are valid.

These statistics can be displayed or printed for the user of Subscriber Section 30 for monitoring transactions, data fields, or combinations of transactions and data field statistics. The comparison of these statistics can then be used for threshold warnings and errors, historical monitoring, projections, and Test Results Report 612, as shown in step 640.

In step 645, Test Result Database 617 stores the Boolean logic and statistical calculations and algorithms employed by PostProcessor 601. In step 650, PostProcessor 601 begins inserting valid transactions into Subscriber's Application Software 620. Any transactions that are not valid are isolated into a separate file. The user of Subscriber Section 30 is notified of these exceptions in Warning/Error Report 627 and/or Test result Report 612. The invalid transactions can subsequently be reviewed by the user of Subscriber Section 30 to create, edit, delete, and save the transactions and or fields. These tasks may be performed within the transaction browsing option of PostProcessor 601.

In an alternative embodiment, PostProcessor 601 begins inserting all valid and invalid transactions into Subscriber's Application Software 620. PostProcessor 601 generates a Warning/Error Report 627 and/or Test Result Report 612 notifying the user of Subscriber Section 30 that warnings and/or errors have been inserted into Subscriber's Application Software 620. These reports provide the user of Subscriber Section 30 with a report detailing each transaction that requires the user of Subscriber Section 30 to review the transaction in Subscriber's Application Software 620 and suggest possible solutions, by means previously described.

In another alternative embodiment, PostProcessor 601 inserts all valid and invalid transactions into Subscriber's

Application Software 620 and does not provide warning and/or error reports.

In yet another alternative embodiment, PostProcessor 601 inserts only valid transactions into Subscriber's Application Software 620 and PostProcessor 601 generates a report of invalid transactions.

In all of these embodiments, PostProcessor 601 inserts the transactions into the file formats required by Subscriber's Application Software 620.

Once the User of Subscriber Section 30 quits PostProcessor 601 and initiates the Subscriber's Application Software 620, all of the transactions will show up in that application. Subscriber Application Software 640 include, but are not limited to relational databases, non-relational databases, and flat file programs. Examples of these databases include Btrieve, Oracle, Sybase, and XBase. Examples of software applications using these databases include Platinum, Great Plains, Macola, Solomon, and Quicken.

The resultant data may be stored by any means known in the art, including, but not limited to hard disks, tapes, magneto-optical disks, compact disks, memory chips. Of course, other forms of data storage are known in the art, and are applicable within the framework of this invention.

After PostProcessor 601 has completed all of its tasks, the end result is that transactions have been automatically entered into Subscriber's Application Software 620 and have been verified and validated, as if the user had manually reviewed, calculated, verified, looked-up and cross referenced, and entered the transactions manually. Use of the invention has made this process much faster, more accurate, and less expensive.

The following is an example scenario of a Subscriber using the invention herein described. In this example, the Subscriber is using accounting software such as Platinum (Platinum Software, Irvine, Calif.) as the Subscriber's Application Software, and the Subscriber wants to automate the processes of data transfer, translation, analysis, and entry for payroll and bank statement reconciliation. The Subscriber, in this example has his payroll processed by a commercial payroll processing company such as Automated Data Processing (ADP) Inc. (Roseland, N.J.). The Subscriber also uses a bank, such as Bank of America (San Francisco, Calif.), for its general checking account. In this example, the payroll processing company and the bank participate in the use the invention herein described, and therefore each have a Provider Section 10 as part of their in-house computer systems, with which they serve as Data Providers 12.

The process begins when the bank and the payroll processing company each use their respective Provider Section 10 computer systems to generate their Provider Data Files 115. Examples of Provider Data Files 115 for payroll and bank statement reconciliation, are shown in Tables 7 and 8, respectively. Once the Provider Data Files 115 have been created, the Data Providers each use their respective Provider Section Telecommunications Software 142 to transmit the data files to Main Processing Section 20.

Once Main Processing Section 20 successfully receives the Provider Data Files 115 from the Provider Section 10, Main Processing Section 20 performs tasks such as determining which subscriber the Provider Data File 115 belongs to, translates the files into MidFormat Files 342, and performing subscriber billing tasks. Examples of the MidFormat Files 342 are shown in Tables 9 and 10. Once these tasks have been completed the MidFormat Files 342 containing the statement information and the payroll information, are placed into the proper Outgoing Data Box 371.

At this point, the subscriber of our example uses Subscriber Telecommunication Software 542 to dial into Main Processing Section 20, and download the two MidFormat Files 342. The subscriber then starts Postprocessor 601 for Payroll. This program performs tasks which include validating, calculating, translating, importing, and reporting. This process is then repeated using PostProcessor 601 for bank statement reconciliation. Examples of Destination Format Files 610 are shown in Tables 11 and 12.

Payroll data has been automatically entered into the accounting software application general ledger, eliminating the need for this data to be entered manually. Subsequently, the subscriber enters the general ledger module of the accounting application software, and prints out an exception report. In this example, the exception report flags the subscriber that a certain employee received and inappropriate amount of money in his last paycheck, and that another certain employee's salary expense was allocated to a non-existent account number.

Bank statement data has been automatically entered into the bank book module of the accounting software application, eliminating the need for this data to be entered manually. The subscriber then enters the bank book module of the accounting software application and prints out reports to examine if any unusual transactions occurred. In this example scenario, a check that does not belong to the subscriber, and a check amount that does not agree with the bank statement amount have been flagged for the subscriber to review.

In summary, the invention herein described eliminates several manual tasks, and substantially reduces the high labor cost associated with these manual tasks. The system is unique for several reasons. First, this invention provides for data exchange between an unlimited number of Data Providers and Subscribers, at a cost several times lower than possible using conventional systems. Second, the system is designed to rapidly connect and configure new Data Providers and Subscribers, as opposed to other systems, which require months or years to implement. Third, the system provides integral means for sophisticated and intelligent data analysis.

In an alternative embodiment, Provider Data File 115 is received at Main Processing Section 20 in the form of a scanned image, such as via a fax transmission. There, text and numeric values are extracted from the fax transmission in a manner to be discussed, and then the resultant data is processed as a Provider Data File or as a MidFormat file according to the methods previously described. Use of this embodiment has the advantage of not requiring an alphanumeric digital (e.g. ASCII) record as a starting point. By the method of this embodiment, a subscriber to the system of this invention needs only a paper Printed Statement 702 with relevant data printed on it, for example, a bank statement, a scanner such as Fax Machine 705 with which to send it to Main Processing Section 20, and a Computer 52 with Modem 54J to serve as Subscriber Section 30, as previously described.

As previously described herein, Provider Data File 115 may any type of data file, including a graphics file. It will be appreciated that the digital data obtained when a scanner scans a printed sheet of paper constitutes one such type of Provider Data File. One can also appreciate that a Subscriber who has been mailed a paper bank statement effectively becomes a Data Provider himself, once he faxes or otherwise provides Main Processing Section 20 with this data. In such a configuration, the Subscriber is the same party as the Data

Provider, since that same party both sends the fax to Main Processing Section 20, and receives back processed data. This patent application refers to such a client site as Unified Provider/Subscriber Section 713. Alternatively, a computer-based machine may send an image bit stream, such as a fax bit stream directly from the computer's memory, without the need to scan from a printed sheet of paper. Alternatively, United Provider/Subscriber Section 713 may send Main Processing Section 20 an alphanumeric file. Although Unified Provider/Subscriber Section 713 may originally receive the paper document or data file from another party (such as a bank, in the case of a bank statement), that original source has no electronic connection with the system of the invention, and hence is not considered to be the Provider in this particular embodiment of the current invention. In yet another alternative embodiment, paper statements or data-containing digital media such as diskettes or tape can be sent to main processing Section 20 by physical means such as mail, courier, or the like; and then directly input into the system such as by scanner or by disc drive or tape drive. Subsequently, the data may be processed in the same manner as if it were received over a telecommunication line.

When a paper document such as a bank statement is transmitted by a conventional fax machine, a graphic file of the printed data is being sent. In other words, the data is sent by transmitting a pictorial representation of the scanned surface's printed and non-printed areas. Generally speaking, forms processing systems perform the following steps: accept an input file in the form of a fax or scanned image of a document, identify the type of document represented, align the image, locate each field of information, extract each field, segment or cut between the characters, perform character recognition, such as by OCR (Optical Character Recognition) and/or ICR (Intelligent Character Recognition), apply outside knowledge ("knowledge engineering") to verify the results, producing an interpretation of the document's image in a form such as an ASCII file. Forms processing software is well known in the art. In U.S. Pat. No. 5,131,053, Bernzotti, et al. describes a system for recognition of characters, and for parsing a page to allow for production of the output characters in a logical sequence. In *The Forms Processing Paradigm Shift* Imaging Magazine, March 1995, pp. 84-95, forty-three such forms processing software-based systems are reviewed. Included in this list is the software by Teraform (San Jose, Calif.), which will be used as the Forms Processing Software 742 in the particular examples to follow. Also, forms processing systems such as GTESS Automated Data Entry System (GTESS Corp., Boulder, Colo.), or Teleform (Cardiff Software, Carlsbad, Calif.) may be used as Forms Processing Software 742 herein.

FIG. 10 shows one embodiment of the invention in which the provider data file is a fax transmission of a paper document sent to Main Processing Section 20 by a subscriber to the system. At Unified Provider/Subscriber Section 713, Printed Statement 702 is placed into Fax Machine 705, and sent to Main Processing Section 20 via a telecommunication link carrying Fax Bitstream 710 between the fax machine and Fax/Modem 780. Once an incoming fax is identified, the computer(s) at Main Processing Section 20 capture each incoming fax page as a unique Fax Bitmap 725

image file. As each Fax Bitmap 725 image file is captured, the resulting image files are put into Fax Queue 715—an incoming fax holding area, such as a hard disk drive directory. Meanwhile, Fax PrepServer A 720 continuously scans Fax Queue 715 for Fax Bitmap 725 image files. Once PrepServer A identifies Fax Bitmap 725 image files in the Fax Queue 715, Fax PrepServer A 720 takes the first Fax Bitmap 725 image file and processes it through the bit map to TIFF Converter 730. Converter 730 is in one embodiment, a TSR program which converts the format of the fax files to a more universally recognized format, such as TIFF (Aldus Corporation, Seattle, Wash.). The first page that is processed may contain information about the fax transmission such as the number of pages transmitted, who the fax transmission came from, who the fax transmission is for, and what type of data is being transmitted: Subsequent Fax Bitmap 725 image files may represent pages of actual transactions. One by one, each Fax Bitmap 725 image file is delivered by Fax PrepServer A 720 to Converter 730, and are processed into TIFF Files 735. After format conversion, Fax PrepServer A 720 creates Fax Reference File 737—a text file containing the names of each TIFF file 735 and the directory location of TIFF files 735. Finally, Fax PrepServer A 720 moves Fax Reference File 737, along with all of the newly created TIFF Files 735 to the next holding location—Forms Processing Input Queue 749.

Fax PrepServer B 740 scans Forms Processing Queue 749 to determine whether any Fax Reference File 737 and TIFF Files 735 are present. Once Fax PrepServer B 740 determines that these are present, it calls the Forms Processing Software 742 to first read the text file and to subsequently process each TIFF file. Forms Processing Software 742 reads the Fax Reference File 737 to determine how many TIFF Files 735 it should process for a given job, and any special handling or processing instructions. Forms Processing Software 742 perform the following steps: accepts input files in the form of TIFF images of documents, identifies the type of document represented (by means such as reading identifying information on the image of a cover page which precedes the transmission of the actual data, or by the recognition of identifying information on the images of actual data pages, such as an account number, name, or a specific form layout), align the image, locate each field of information, extract each field, segment or cut between the characters, perform character recognition, such as by OCR (Optical Character Recognition) and/or ICR (Intelligent Character Recognition), apply outside knowledge ("knowledge engineering") to verify the results. The resultant output file is a alphanumeric text file such as Comma-Delimited ASCII File 770, is placed into MPS Incoming Data Box 355, and the original TIFF files are placed in Forms History Directory 775 for archival and backup purposes. The newly created Comma-Delimited ASCII File 770 is similar in structure and content to Provider Data Files 115 previously described with respect to other embodiments of the current invention, and hence may be further processed according to the same methods as described with respect to FIGS. 6A, 6B, 6C, 7, 8, and 9. Alternatively, Forms Processing Software 742 can place its output directly into a designated MidFormat, thus skipping one format conversion step in the system process.

In an example of the invention in use, we will assume that Clemco wants their bank statements automatically entered into their Platinum accounting module. Having received the printed statement in the mail from their bank, Clemco puts the document into a fax machine, and sends it in an ordinary manner to the telephone number of Main Processing Section 20. The fax is turned into an MidFormat File 342 according to the methods described with respect to FIG. 10. MidFormat File 342 is then sent back to Clemco by modem, and processed into Subscriber Application Software 620 as illustrated in FIGS. 6A, 6B, 6C, 7, 8, and 9. Consequently, Clemco now has in their Platinum accounting module the data with was formerly available to them only on a printed sheet of paper. Furthermore, the data has been validated.

In an alternative embodiment, after a file is converted to a MidFormat File 342, the file is compressed and added to a database in Telecommunications Software 542. While Telecommunications Link 21 is active, a program is run to send an electronic mail message to the user that the file is processed and ready for download. The file is attached to the mail message so that when the user reads the message, the file is automatically downloaded.

Data manipulations in the current invention may be metaphorically described as "manufacturing assembly line" style. The spirit of this feature of the invention is to move "raw data" into place, convert the "raw data" into usable data via processing stages, and finally move the "finished data" out for subscribers to use. Such an approach is useful because of the need to handle various pieces of raw and partially processed data with different programs created by, or associated with the present invention. The process includes an "assembly line" and "robots". The "assembly line" are the devices (both hardware and software) that move the data from one processing stage to the next processing stage. The "robots" are the programs comprising the current invention.

All Main Processing Section 20 functions in this and other embodiments of the invention can, if desired, be conveniently carried out on a single telecommunications-capable computer, as well as with a task-distributed network.

MidFormat, Provider Format, or Destination Format files may be, in practice, any format desired, including standard EDI formats.

A portion of the disclosure of this patent contains material which is subject to copyright protection. In particular certain of the file formats, including but not limited to the Midformat file format, are subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

The method and apparatus of the present invention are described above with reference to specific embodiments. Other embodiments will be apparent to those of ordinary skill in the art. It is therefore not intended that this invention be limited except as indicated by the appended claims.

TABLE 1

(© 1994 AutoEntry Online, Inc.)

CUSTOMER	Master Customer Definition Table
Customer ID	6 Alphanumeric *Drives internal AR billing system (Platinum AR)
Customer Name	40 Alphanumeric
Contact Person	30 Alphanumeric
Address 1	30 Alphanumeric
Address 2	30 Alphanumeric
City	40 Alphanumeric
State	2 Alphanumeric
Country	15 Alphanumeric
Zip Code	10 Alphanumeric
Voice Number	10 Alphanumeric
Fax Number	10 Alphanumeric
Credit Card Name	20 Alphanumeric
Credit Card Number	20 Alphanumeric
Credit Card Expir Date	4 Alphanumeric
Credit Card Holder Name	30 Alphanumeric

TABLE 2

(© 1994 AutoEntry Online, Inc.)

MAILBOX	Master Mailbox Definition Table
Mailbox ID	11 Alphanumeric *
Customer ID	6 Alphanumeric
File Server	47 Alphanumeric
Disk Volume	15 Alphanumeric
Directory Path	80 Alphanumeric
Box Type	1 Alphanumeric Source(S) or Target(T)
PreOrPost Processor	12 Alphanumeric

TABLE 3

(© 1994 AutoEntry Online, Inc.)

USAGE	Transactions Table for Tracking and Accounting	
Batch ID	12 Alphanumeric	
Customer ID	6 Alphanumeric	
Provider ID	6 Alphanumeric	
Data Import Date	6 Date	NOTE: IN PARADOX, DATE AND NUMERIC FIELDS DO NOT ACTUALLY USE THE FIELD SIZE; INCLUDED FOR CONSISTENCY.
Data Import Time	8 Alphanumeric	
Transmission Status	20 Alphanumeric	
Data Transmission Date	6 Date	
Data Transmission Time	8 Numeric	
Target Data Module	2 Alphanumeric	
Number of Headers	8 Numeric	
Number of Line Items	8 Numeric	Time (A8) is "HH:MM:SS"

31

TABLE 4

(© 1994 AutoEntry Online, Inc.)		
SYS_LOG	System Log File - Used by PrepServer and Overlays	
Entry Date	6 Date	*
Entry Time	8 Alphanumeric	*Time (A8) is "HH:MM:SS"
Comment	80 Alphanumeric	

TABLE 5

(© 1994 AutoEntry Online, Inc.)		
BATCHNUM	Incrementing Batch ID Number- Only 1 record and 1 field	
Last Batch Number	8 Numeric	

TABLE 6

(© 1994 AutoEntry Online, Inc.)		
PreProcessor Command-Line Arguments		
Here is a sample usage message put out by the PP, describing the arguments it can use.		
pp01 ver 1.0 -- ADP PreProcessor for Subscriber Smith Co.		
usage: pp01 [-log logfile] [-verbose] [-report] [-usetabs]		
datafile (work dir)		
where: log	= name of alternate system logfile	
verbose	= output verbose statements	
report	= output human readable report	
usetabs	= output spreadsheet ready report	

32

TABLE 6-continued

(© 1994 AutoEntry Online, Inc.)

PreProcessor Command-Line Arguments

datafile	= Raw Transaction File (RTF) to translate
work dir	= Directory to process RTF in.

Note:

"pp01" Should be replaced with the name of the PP.

"ver 1.0" Should be replaced with the appropriate version.

"ADP" Should be replaced with the actual provider name the PP is designed for.

"Smith Co." Should be replaced with the actual Subscriber name the PP finds in the RTF.

TABLE 7

(© 1994 AutoEntry Online, Inc.)		
Example Payroll File Layout from Data Provider		
20	8xxxxMAR3194032694909	
	8xxxx 1021-0001PR03314-0019106664	NET PAY
	8xxxx 1021-0001PR03314+0000000003	NET PAY
	8xxxx 1140-0001PR03314-0000047028	DENTAL INSURANCE
	8xxxx 1150-0001PR03314-0000052000	EMPLOYEE LOANS
	8xxxx 1900-0001PR03314-0000349241	EMPLOYEE
25	ADVANCE	

TABLE 8

(© 1994 AutoEntry Online, Inc.)		
Example Bank File Layout from Data Provider		
30	319761,02,02,93,10141010,Employee Group Dental,000817.00	
	319777,02,02,93,10141010,Employee Group Medical,000103.69	
	319735,02,02,93,10141010,Employee Group WAS,000152.38	
	319741,02,02,93,10141010,Supplemental WAS,000092.35	
	036565,03,03,93,10211020,Employer Group Dental,000012.00	

TABLE 9

```
(© 1994 AutoEntry Online, Inc.)
Example Payroll Midformat File Layout
```

```
Subj:  ADP midformat data
Date:  94-08-22 23:28:00 EDT
From:  Santa Clara ADP
To:    Clemco
# Clemco Mid-Format data file
#
# The standardized midformat should have the
# following characteristics:
#
# Field Name           Field Length
#
# Customer Key         10
# Account Number       32
# Payroll Date         MM/DD/YY
# Debit/Credit         5
# Description           15
..001", "1021-0001", "03/31/94", "-", "191066.64", "NET PAY"
..001", "1021-0001", "03/31/94", "-", "0.03", "NET PAY"
..001", "1140-0001", "03/31/94", "-", "470.28", "DENTAL INSURANCE"
..001", "1150-0001", "03/31/94", "-", "520.00", "EMPLOYEE LOANS"
..001", "1900-0001", "03/31/94", "-", "3492.41", "EMPLOYEE ADVANCE"
```

TABLE 10

(© 1994 AutoEntry Online, Inc.) Example Bank Midformat File Layout	
Subj: ADP midformat data	5
Date: 94-08-22 23:28:00 EDT	
From: Santa Clara ADP	
To: Clemco	
# Clemco Mid-Format data file	
#	
# The standardized midformat should have the	10
# following characteristics:	
#	
# Field Name Field Length	
# Customer Key 10	
# Check Number 10	
# Check Date MM/DD/YY	15
# Check Transit Number 15	
# Check Amount 15	
,"005","319761","02/02/93","10141010","817.00","	
,"005","319777","02/02/93","10141010","103.69","	
,"005","319735","02/02/93","10141010","152.38","	
,"005","319741","02/02/93","10141010","92.35","	20
,"005","036565","03/03/93","10211020","12.00","	

TABLE 11

(© 1994 AutoEntry Online, Inc.) Example Payroll Destination File Layout	
1021,"0001","IP","NET PAY","03/31/94","03/31/94","3767","3","191066.65","	
1021,"0001","IP","NET PAY","03/31/94","03/31/94","3767","3","0.03","	
1140,"0001","IP","DENTAL INSURANCE","03/31/94","03/31/94","3767","3","470.28","	
1150,"0001","IP","EMPLOYEE LOAN","03/31/94","03/31/94","3767","3","520.00","	
1900,"0001","IP","EMPLOYEE ADV","03/31/94","03/31/94","3767","3","3492.41","	

TABLE 12

(© 1994 AutoEntry Online, Inc.) Example Bank Destination File Layout	
,"319761","2","01/15/93","1000","0100","817.00","817.00","Y","Y","02/02/93","	
,"319777","2","01/15/93","1000","0100","103.69","103.69","Y","Y","02/02/93","	
,"319735","2","01/15/93","1000","0100","152.38","152.38","Y","Y","02/02/93","	
,"319741","2","01/15/93","1000","0100","92.35","92.35","Y","Y","02/02/93","	
,"036565","2","02/15/93","1000","0100","12.00","12.00","Y","Y","03/03/93","	

What is claimed:

1. In a distributed computer system including a data provider section, a main processing section, and a data subscriber section, each said section selectably coupleable to another section by a communications link including communications control means, a method for translating and transferring data from a provider source format to a subscriber destination format using a priori knowledge of said source and destination formats between said data provider section and a said data subscriber section, said method comprising the steps of:

(A) transmitting, by said provider section, a particular provider data file from said provider section to said main processing section via said communications link, said transmitting including the steps of:

monitoring communications from said provider section;

recognizing receipt of a valid communications initiation prompt generated by said provider section communications control means according to predetermined communications initiation rules, said rules

providing safeguards for ensuring the quality of the data delivered to valid data subscribers;

initiating connection of provider section with main processing section in response to receipt of said valid communications initiation prompt;

connecting said provider section with said main processing section and upon being validated as a valid provider section, receiving clearance from said main processing section to transmit a copy of said provider data file to said main processing section;

reading said provider data file;

transmitting said provider data files over said communications link to said main processing section; and

determining whether said provider data file has been transmitted successfully and signaling successful data transmission to said main processing section; and

(B) receiving, by said main processing section, said provider data file from said provider section and storing said received data files in said main processing section;

(C) translating, by said main processing section, said stored provider data file by: (i) copying said stored provider data file to a work area file within said main

processor section, and (ii) converting said work area file from the native format of said provider data file to a midformat file format;

(D) forming a communications link, by said main processing section, with a particular one of said subscriber sections and transmitting said midformat file to said subscriber section;

(E) receiving, by subscriber section, said midformat file from said main processing section and storing said received file in storage within said subscriber section; and

(F) translating using said selected second-to-third file translation procedure, by said subscriber section, said midformat file into a subscriber destination format file compatible with said preselected subscriber application software routine.

2. The method in claim 1, wherein said step (A) of transmitting, further includes, prior to said step of monitoring, the steps of:

35

appending a subscriber destination address to said provider data file uniquely representative of a specific subscriber that the data in said particular data file is intended to ultimately reach; and

loading said provider data file with said appended destination address into an outgoing data box associated with said provider within said provider section.

3. The method in claim 2, wherein said step (A) of transmitting, further includes, after said step of determining whether said provider data file has been transmitted successfully and signaling successful data transmission to said main processing section, the steps of:

in response to receipt of said successful data transmission signal, moving the original provider data file from said outgoing data box to a history data box within storage means of said provider section; and

in response to an indication that said data transmission to said main processing section was not successful, writing an error indication in a provider section error log.

4. The method in claim 1, wherein said provider data file format is a digital image data bit stream encoding a printed document.

5. The method in claim 4, wherein said data provider section comprises a facsimile machine including an optical scanner for generating a graphical image format file of a printed document, and wherein said provider file is communicated to said main processing section via facsimile transmission of said graphical image file.

6. The method in claim 4, wherein said digital image data bit stream is generated directly from information stored in memory in a digital computer.

7. The method in claim 4, wherein said data provider section and said data subscriber section are the same entity, and wherein said entity facsimile transmits provider data in the form of a scanned printed statement to said main processing section and receives a subscriber format file in return from said main processing section.

8. The method in claim 4, further comprising the steps of: processing said graphic data file using a form processing procedure to generate a non-graphical digital data file including information extracted from said scanned graphical data.

9. The method in claim 4, wherein said form processing procedure includes the steps of:

accepting an input file in the form of a graphical image of a document; identifying the type of document represented from among a plurality of known possible document types; aligning the graphical image with known document type fields; locating each field of information; extracting each field; segmenting between the characters in each said field; performing character recognition; and producing an interpretation of the document's image including alphanumeric components in digital form.

10. The method in claim 1, wherein said communication channel between said provider section and said main processing section comprises physical transportation of the data containing media from a location of a provider to a location of said main processing section; conversion of said data containing media into electronic form; and input of said electronic form to said main processing section.

11. The method in claim 10, wherein said data containing media is selected from the group consisting of a paper statement, a data-containing diskette, a data-containing tape, and combinations thereof.

12. The method in claim 11, wherein said physical communications is selected from the group consisting of: mail, courier, and combinations thereof.

36

13. In a distributed computer system including a data provider section, a main processing section, and a data subscriber section, each said section selectably couplable to another section by a communications link including communications control means, a method for translating and transferring data from a provider source format to a subscriber destination format using a priori knowledge of said source and destination formats between said data provider section and a said data subscriber section, said method comprising the steps of:

(A) transmitting, by said provider section, a particular provider data file from said provider section to said main processing section via said communications link;

(B) receiving, by said main processing section, said provider data file from said provider section and storing said received data files in said main processing section, said step of receiving including the steps of:

monitoring a communication link port for messages from a calling party requesting a communication connection according to predetermined monitoring rules and determining whether a provider section is calling main processing section requesting a communication connection;

determining whether to allow said calling party to connect to said main processing section based upon predetermined security information stored in said computer system and security access rules;

disconnecting from an invalid calling party not satisfying said security access rules; and

maintaining said communication connection between a valid provider section and a valid main processing section satisfying said security access rules and granting access between said valid provider section and said valid main processing section; and

(C) translating, by said main processing section, said stored provider data file by: (i) copying said stored provider data file to a work area file within said main processor section, and (ii) converting said work area file from the native format of said provider data file to a midformat file format;

(D) forming a communications link, by said main processing section, with a particular one of said subscriber sections and transmitting said midformat file to said subscriber section;

(E) receiving, by subscriber section, said midformat file from said main processing section and storing said received file in storage within said subscriber section; and

(F) translating using said selected second-to-third file translation procedure, by said subscriber section, said midformat file into a subscriber destination format file compatible with said preselected subscriber application software routine.

14. The method in claim 13, wherein said step of determining whether to allow said calling party to connect within step (B), comprises the steps of:

activating a billing and administrative system database storing information regarding all valid providers including each provider's respective provider section, and corresponding passwords and security levels; and accesses said billing and administrative system database to lookup and determine whether the calling party is a valid provider section.

15. The method in claim 14, wherein said step (B) of receiving said provider data file from said provider section further comprises, after said step of maintaining said communication connection, the steps of:

responding to a provider section request that said main processing section receive a particular provider data file;

determining which provider incoming data box should be used based on predetermined provider criteria and storing said particular provider data file to said incoming data box; and

sending a data transmission receipt to said provider section informing said provider section that said particular provider data file transmitted successfully.

16. In a distributed computer system including a data provider section, a main processing section, and a data subscriber section, each said section selectably couplable to another section by a communications link including communications control means, a method for translating and transferring data from a provider source format to a subscriber destination format using a priori knowledge of said source and destination formats between said data provider section and a said data subscriber section, said method comprising the steps of:

(A) transmitting, by said provider section, a particular provider data file from said provider section to said main processing section via said communications link;

(B) receiving, by said main processing section, said provider data file from said provider section and storing said received data files in said main processing section;

(C) translating, by said main processing section, said stored provider data file by: (i) copying said stored provider data file to a work area file within said main processor section, and (ii) converting said work area file from the native format of said provider data file to a midformat file format;

(D) forming a communications link, by said main processing section, with a particular one of said subscriber sections and transmitting said midformat file to said subscriber section, said step of forming a communication link further comprising the steps of:

scanning a communication link port for a calling party attempting to communicate with said main processor section according to predetermined rules;

determining whether to allow calling party to communicate with said main processing section based upon security information and security access rules;

disconnecting from a calling party determined to be invalid based on said security information and security access rules; and

maintaining said communication connection between a valid subscriber section and a valid main processing section and granting subscriber section access between said valid subscriber section and said valid main processing section; and

accepting a request for data from said subscriber section and transmitting said requested data in said midformat file in response thereto; and

(E) receiving, by subscriber section, said midformat file from said main processing section and storing said received file in storage within said subscriber section; and

(F) translating using said selected second-to-third file translation procedure, by said subscriber section, said midformat file into a subscriber destination format file compatible with said preselected subscriber application software routine.

17. The method in claim 16, wherein said, step (D) of forming said communications link further comprises, after disconnecting from said invalid calling party, the step of

recording information concerning the invalid communication attempt including a date-stamp and time-stamp in a system database.

18. In a distributed computer system including a data provider section, a main processing section, and a data subscriber section, each said section selectably couplable to another section by a communications link including communications control means, a method for translating and transferring data from a provider source format to a subscriber destination format using a priori knowledge of said source and destination formats between said data provider section and a said data subscriber section, said method comprising the steps of:

(A) transmitting, by said provider section, a particular provider data file from said provider section to said main processing section via said communications link;

(B) receiving, by said main processing section, said provider data file from said provider section and storing said received data files in said main processing section;

(C) translating, by said main processing section, said stored provider data file by: (i) copying said stored provider data file to a work area file within said main processor section, and (ii) converting said work area file from the native format of said provider data file to a midformat file format;

(D) forming a communications link, by said main processing section, with a particular one of said subscriber sections and transmitting said midformat file to said subscriber section;

(E) receiving, by subscriber section, said midformat file from said main processing section and storing said received file in storage within said subscriber section; and

(F) translating using said selected second-to-third file translation procedure, by said subscriber section, said midformat file into a subscriber destination format file compatible with said preselected subscriber application software routine, said step (F) of translating further including the steps of:

selecting a particular post-processor based on predetermined post-processor selection rules including the identity of said requesting subscriber section and the identity of said preselected application software routine;

searching for a particular midformat file from among available midformat files based on predetermined search criteria;

reading said midformat file data into a memory of said subscriber section;

reformatting the data from said midformat file into the destination format required by subscriber's application software according to predetermined data manipulation routines; and

inserting said data into said subscriber application software.

19. The method in claim 18, further comprising, after the step of reformatting said data from said midformat file, the steps of:

testing said reformatted data in the translated destination format to validate the data; and if said testing indicates that an error has occurred during said reformatting, then issuing an error condition message to the user to indicate an error condition.

20. In a computer system including a data provider section, a main processing section, and a data subscriber section, each said section selectably couplable to another

39

section by a communications channel including communications control means, a method for translating and transferring data from a native provider source format to a subscriber destination format using a priori knowledge of said source and destination file formats between said data provider section and a said data subscriber section, said method comprising the steps of:

- (A) transmitting a provider data file, formatted in a particular provider data format from among a plurality of possible provider data formats, from said provider section to said main processing section via said communications channel;
- (B) receiving said provider data file from said provider section by said main processing section;
- (C) selecting one particular provider-to-midformat file format translation procedure, from among a plurality of possible provider-to-midformat translation procedures, for translating said particular provider source format into a particular midformat based only on a priori knowledge of the identity of said provider and a file type characteristic of said particular provider file;
- (D) translating said received provider data file from said provider source format to said midformat file using said selected one particular provider-to-midformat translation procedure without iterative trial fitting of said provider data into said midformat file;
- (E) communicating said file in said midformat file from said main processing section to said subscriber section;
- (F) selecting a particular midformat-to-subscriber format file translation procedure from among a plurality of possible midformat-to-subscriber format file translation procedures for translating said particular midformat file into a particular subscriber format based only on a priori knowledge of the identity of said subscriber destination and a type characteristics of said subscriber file; and
- (G) translating, using said selected midformat-to-subscriber format file translation procedure, said midformat file into said subscriber destination format file compatible with said preselected subscriber application software routine.

21. The method in claim 20, wherein said step (A) of transmitting provider data files from said provider section to said main processing section via said communications channel further comprises the steps of:

- monitoring communications from said provider section;
- recognizing receipt of a valid communications initiation prompt generated by said provider section communications control means according to predetermined communications initiation rules, said rules providing safeguards for ensuring the quality of the data delivered to valid data subscribers;

initiating connection of provider section with main processing section in response to receipt of said valid communications initiation prompt;

40

connecting said provider section with said main processing section and upon being validated as a valid provider section, receiving clearance from said main processing section to transmit a copy of said provider data file to said main processing section;

reading said provider data files;

transmitting said provider data files over said communications link to said main processing section; and

signaling successful data transmission to said main processing section.

22. The method in claim 20, wherein said computer system is a distributed computer system.

23. The method in claim 20, wherein said provider data file format is a digital image data bit stream encoding a printed document.

24. The method in claim 23, wherein said data provider section comprises a facsimile machine including an optical scanner for generating a graphical image format file of a printed document, and wherein said provider file is communicated to said main processing section via facsimile transmission of said graphical image file.

25. The method in claim 23, wherein said data provider section and said data subscriber section are the same entity, and wherein said entity facsimile transmits provider data in the form of a scanned printed statement to said main processing section and receives a subscriber format file in return from said main processing section.

26. The method in claim 23, wherein said communication channel between said provider section and said main processing section comprises physical transportation of the data containing media from a location of a provider to a location of said main processing section; conversion of said data containing media into electronic form; and input of said electronic form to said main processing section.

27. The method in claim 23, further comprising the steps of:

- processing said graphic data file using a form processing procedure to generate a non-graphical digital data file including information extracted from said scanned graphical data.

28. The method in claim 27, wherein said form processing procedure includes the steps of:

- accepting an input file in the form of a graphical image of a document; identifying the type of document represented from among a plurality of known possible document types; aligning the graphical image with known document type fields; locating each field of information; extracting each field; segmenting between the characters in each said field; performing character recognition; and producing an interpretation of the document's image including alphanumeric components in digital form.

* * * * *



US005469563A

United States Patent [19]

Morita

[11] Patent Number: 5,469,563

[45] Date of Patent: Nov. 21, 1995

[54] METHOD AND CONTROL APPARATUS FOR SELF DIAGNOSIS

[75] Inventor: Shinya Morita, Kanagawa, Japan

[73] Assignee: Sony Corporation, Tokyo, Japan

[21] Appl. No.: 101,198

[22] Filed: Aug. 3, 1993

[30] Foreign Application Priority Data

Aug. 3, 1992 [JP] Japan 4-224522

[51] Int. Cl.⁶ G06F 11/34

[52] U.S. Cl. 395/183.01; 364/267; 364/DIG. 1; 379/1; 395/183.06

[58] Field of Search 371/15.1, 25.1, 371/26; 364/267, 267.4, 267.8; 379/1; 355/203; 340/853.2; 395/575

[56] References Cited

U.S. PATENT DOCUMENTS

4,814,969 3/1989 Kiyooka .
 4,931,963 6/1990 Kimura et al. .
 5,130,936 7/1992 Sheppard et al. .

FOREIGN PATENT DOCUMENTS

0207175 1/1987 European Pat. Off. .
 0476680 3/1992 European Pat. Off. .
 2580091 10/1986 France .

598012 1/1984 Japan G05B 23/02
 184952 10/1984 Japan G06F 11/12
 251860 11/1987 Japan .
 211133 8/1989 Japan G05B 23/02

OTHER PUBLICATIONS

SMPTE Journal, vol. 93, No. 2, Feb. 1984, USA pp. 149-154 M. G. Rose 7 G. Warren 'Diagnostics for a Microprocessor-Based Videotape Recorder'.

Primary Examiner—Robert W. Beausoliel, Jr.

Assistant Examiner—Albert Décady

Attorney, Agent, or Firm—William S. Frommer; Alvin Sinderbrand

[57] ABSTRACT

A self diagnosis method and control apparatus for positive self diagnosis of equipment having complicatedly interlaced mechanical portions. Equipment under diagnosis is divided into a plurality of areas under diagnosis, each of which is assigned an evaluation variable. Diagnosis is executed in units of diagnostic steps classified for each area under diagnosis. Based on an execution result, points are allotted to the evaluation variable. Based on the evaluation variable, or a sum of allotted points, each area under diagnosis is diagnosed and a diagnostic result is displayed for self diagnosis. Meanwhile, those diagnostic steps which are determined invalid by an execution result of another diagnostic step will not be executed. The equipment under diagnosis includes those which contain mechanical portions.

20 Claims, 11 Drawing Sheets

Self Diagnosis

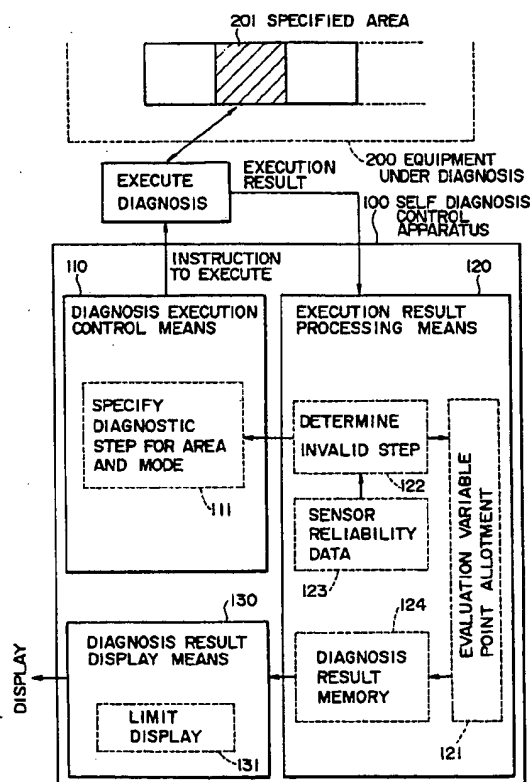


FIG. 1

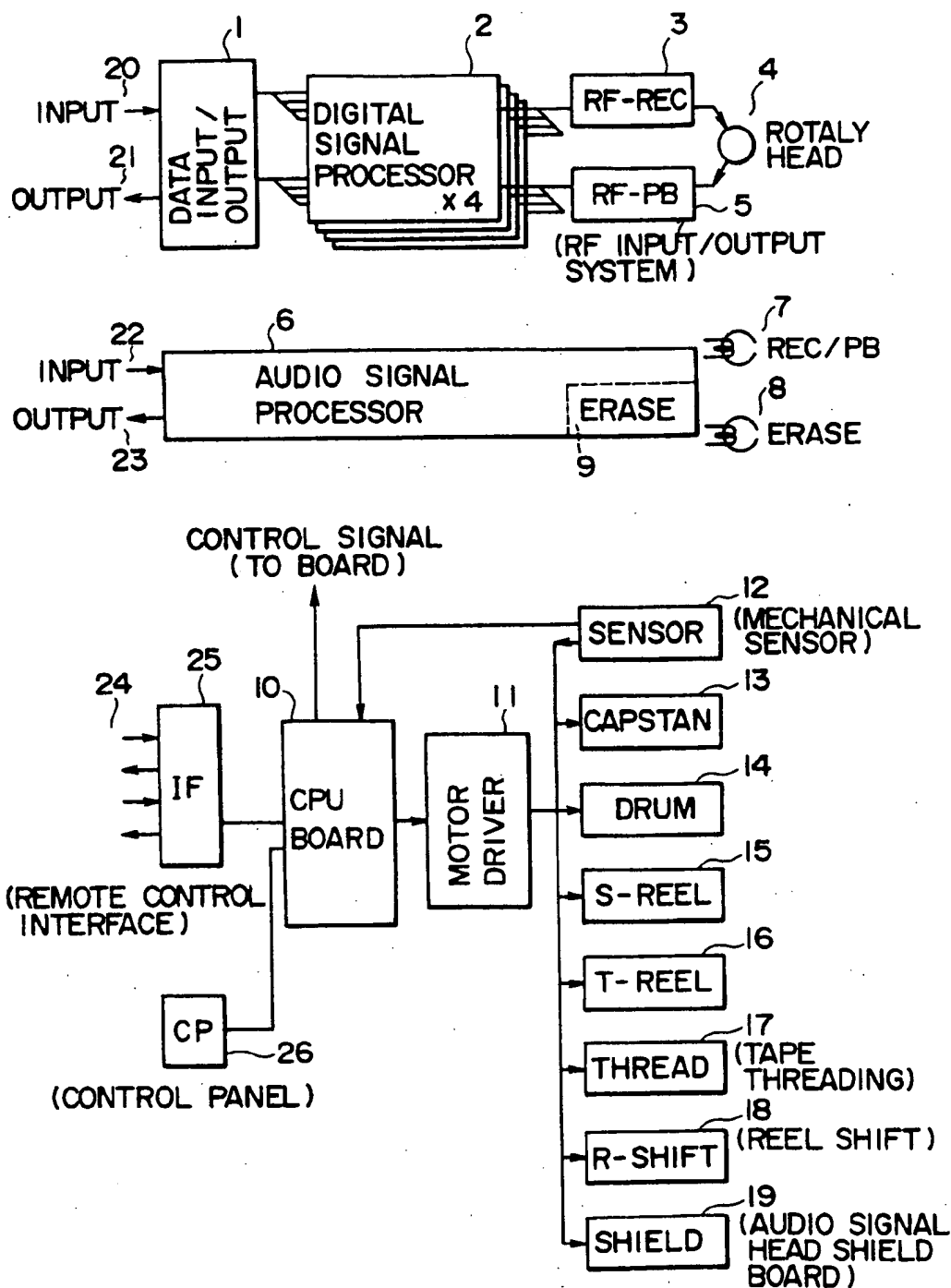


FIG. 2

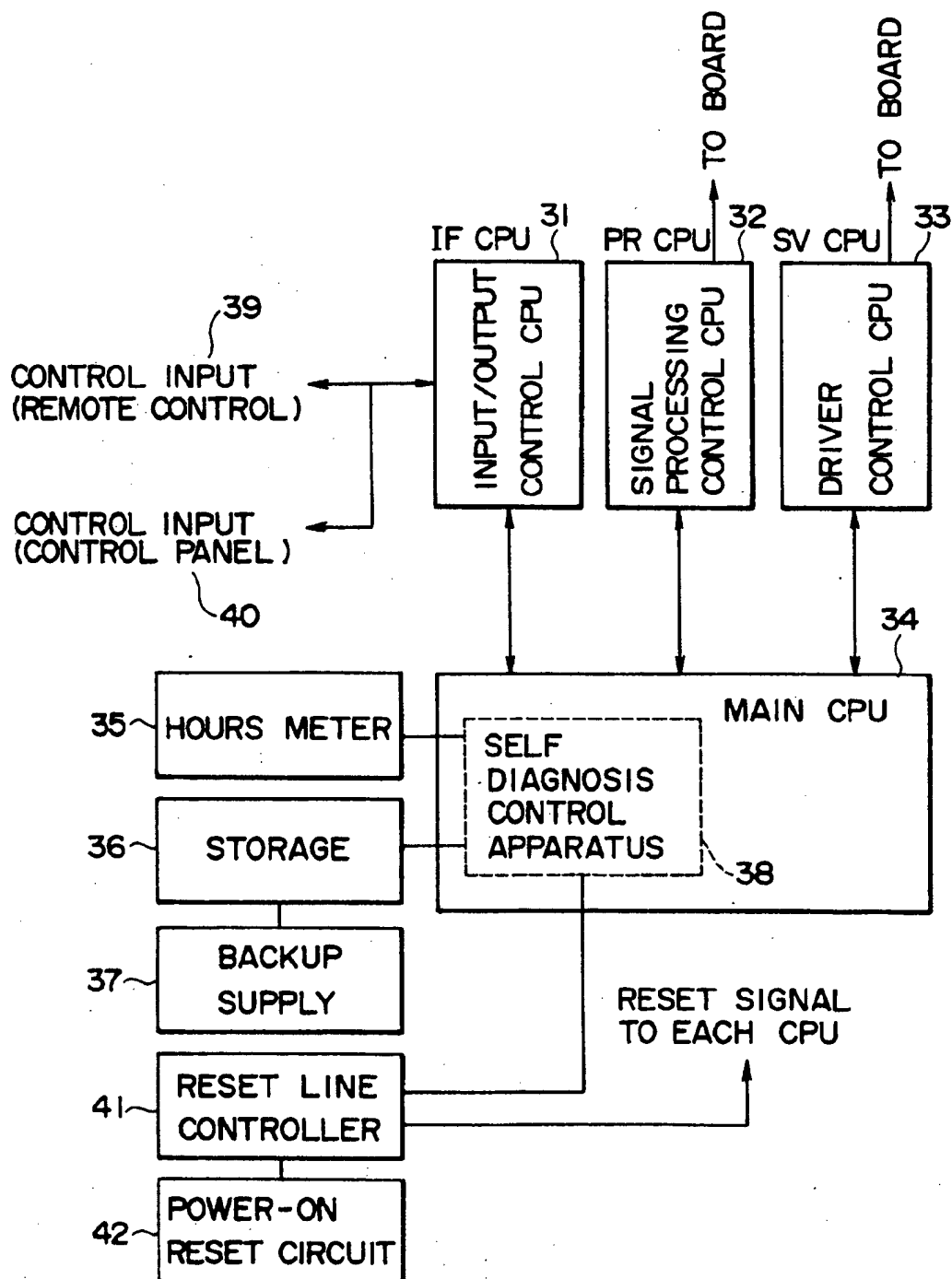


FIG. 3

SELF DIAGNOSIS CAPABILITY

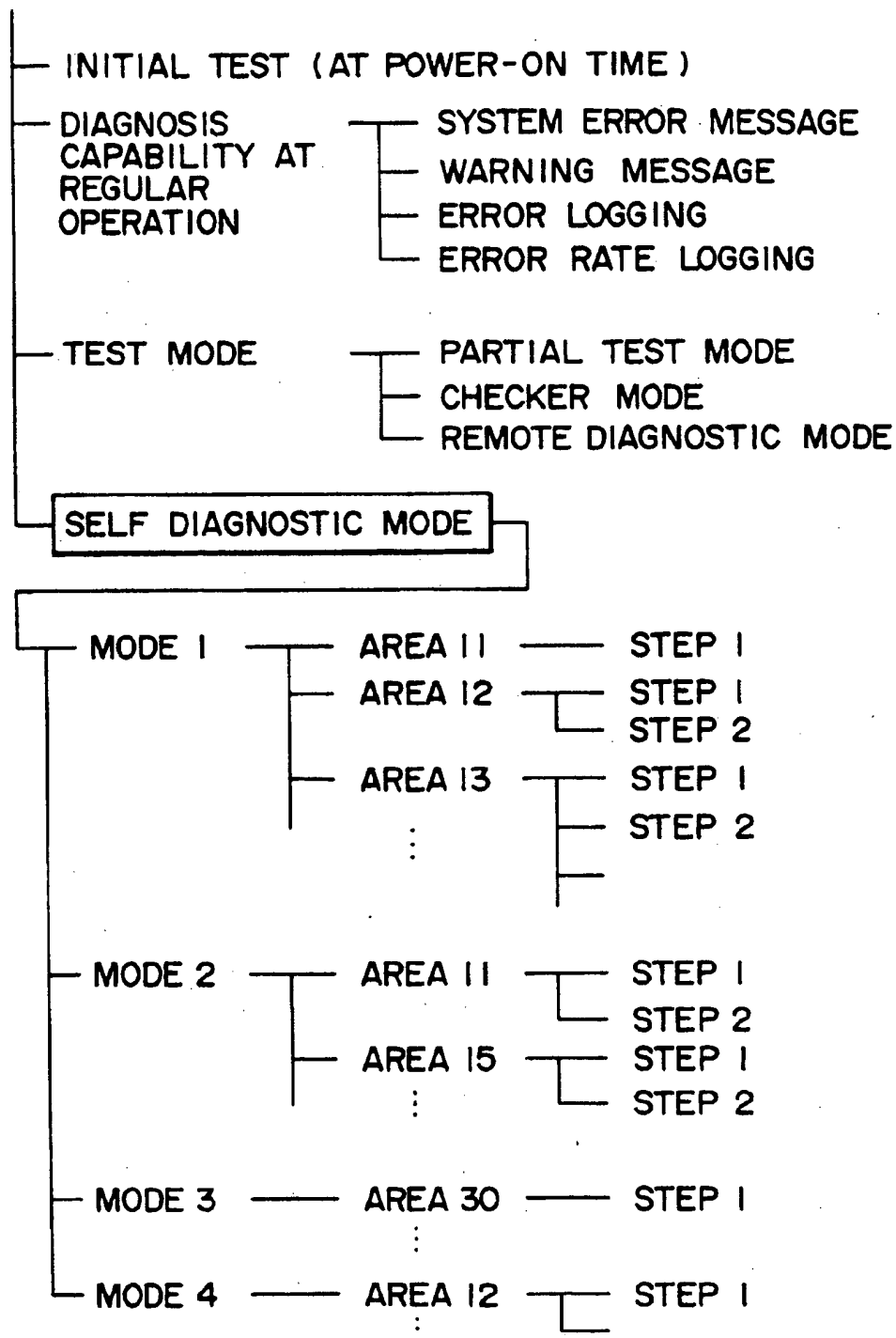


FIG. 4

AREA No.

10	proc_all	(SIGNAL PROCESSING SYSTEM)
11	PIF	DATA INPUT/OUTPUT BOARD
12	PR (1)	DIGITAL SIGNAL PROCESSING (1) BOARD
13	PR (2)	DIGITAL SIGNAL PROCESSING (2) BOARD
14	PR (3)	DIGITAL SIGNAL PROCESSING (3) BOARD
15	PR (4)	DIGITAL SIGNAL PROCESSING (4) BOARD
16	RF_REC	RF RECORDING SYSTEM
17	RF_PB	RF PLAYBACK SYSTEM
20	AE_all	(AUDIO SIGNAL PROCESSING SYSTEM)
21	ID	AUDIO SIGNAL ID RECORD/PLAYBACK SYSTEM
22	AE	AUDIO SIGNAL RECORD/PLAYBACK SYSTEM
23	ERASE	ERASE SYSTEM
30	cont_all	(CONTROL SYSTEM)
31	SSP	CPU BOARD
32	MD	MOTOR DRIVE BOARD
33	IF	REMOTE INPUT/OUTPUT BOARD
34	DM	DATA MODULE BOARD
35	CP	CONTROL PANEL
40	path_all	(TAPE TRAVEL SYSTEM)
41	Drum	ROTARY HEAD SYSTEM
42	S_Reel	SUPPLY REEL MOTOR SYSTEM
43	T_Reel	TAKE-UP REEL MOTOR SYSTEM
44	Capstan	CAPSTAN MOTOR SYSTEM
45	Tension	TENSION CONTROL SYSTEM
50	load_all	(LOADING SYSTEM)
51	Thread	THREAD SYSTEM
52	CassCon	CASSETTE COMPARTMENT SYSTEM
53	R_Shift	REEL SHIFT SYSTEM
54	Shield	SHIELD BOARD UP/DOWN SYSTEM
55	Sensor	SENSORS
80	misc	(MISCELLANEOUS)
81	PS	POWER SUPPLY
82	FAN	COOLING FAN
83	mother 1	MAIN MOTHER BOARD
84	mother 2	SUB MOTHER BOARD
85	data_in	DATA INPUT
86	data_out	DATA OUTPUT
87	remote	REMOTE INPUT/OUTPUT
89	tape	CASSETTE TAPE
90	overall	(OVERALL OPERATION)
91	EE_all	
92	PB_all	
93	REC_all	

FIG. 5

- (1) TURN ON POWER AND REMOVE UNNECESSARY CONNECTIONS AND TAPE.
↓
- (2) SELECT A DESIRED MODE ON THE CONTROL PANEL (OR FROM THE REMOTE CONTROL).
(NORMALLY, SELECT "ALL".)
↓
- (3) IF A SPECIFIC AREA ALONE IS TO BE DIAGNOSED, ENTER THAT AREA.
(NORMALLY, SELECT "ALL".)
↓
- (4) START THE SELF DIAGNOSTIC MODE.
↓
- (5) FOLLOW AN INSTRUCTION TO LOAD A SPECIFIC TAPE AS REQUIRED.
↓
- (6) WHEN THE DIAGNOSIS HAS BEEN COMPLETED, READ A RESULT MESSAGE BY OPERATING THE RESULT DISPLAY MENU ON THE CONTROL PANEL.
↓
- (7) IF THE MESSAGE TELLS AN ABNORMAL CONDITION, MAKE TROUBLESHOOTING ACCORDINGLY AND PERFORM THE DIAGNOSIS ALL OVER AGAIN.
↓
- (8) IF NO ABNORMAL CONDITION PERSISTS, END THE DIAGNOSIS.

FIG. 6A

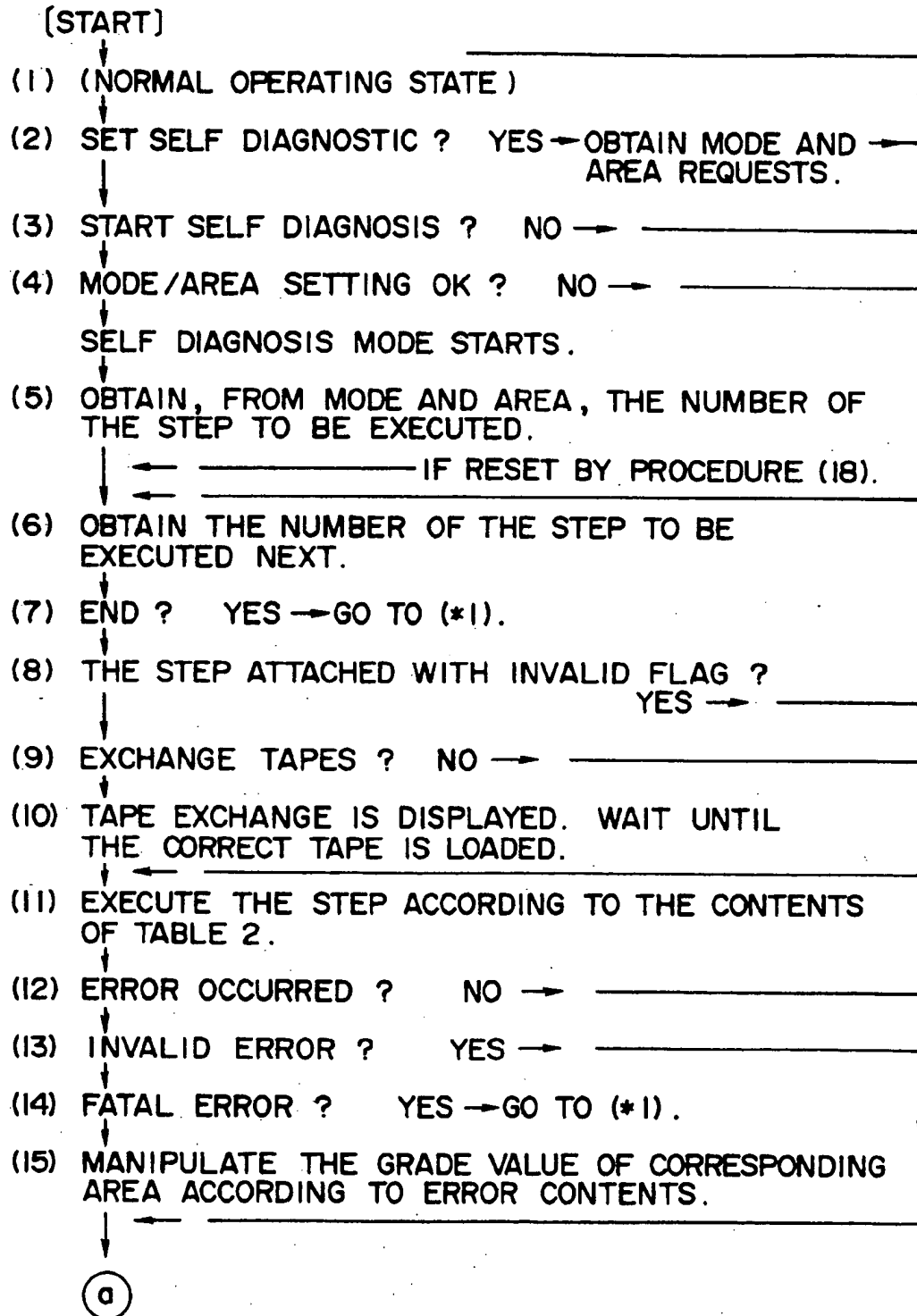


FIG. 6B

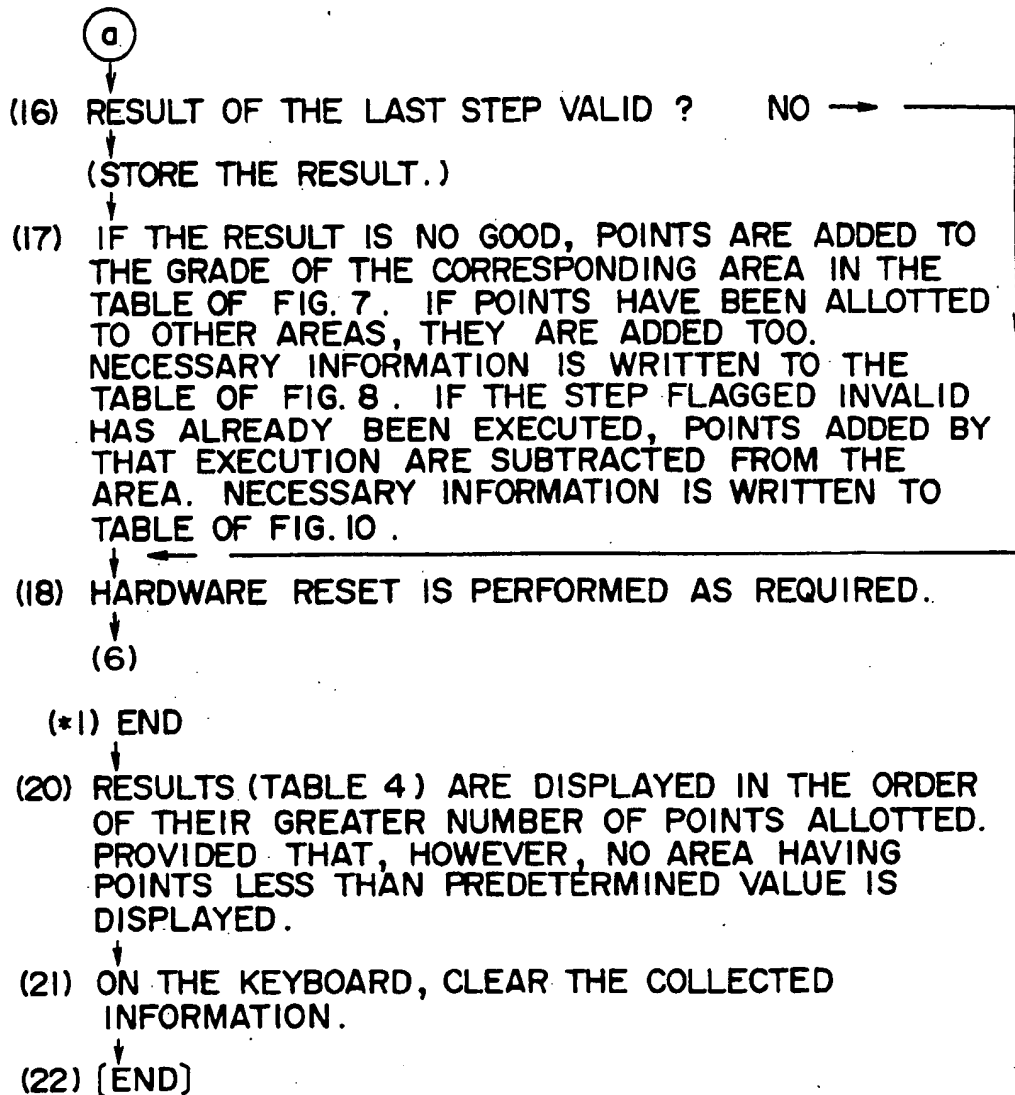


FIG. 6

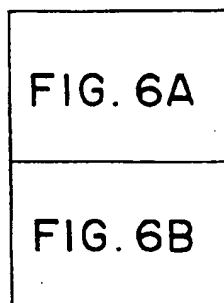


FIG. 7

MODE	1	2	3	4	TOTAL NUMBER OF STEPS
TOTAL NUMBER OF STEPS	30	20	1	5	
AREA (11) (PIF)	1	3 4			3 ← (*1)
AREA (12) (PR(1))	1 2			5 6	4
AREA (13) (PR(2))	7 8			9 10	4

↑
(*2)

FIG. 8

STEP No.	(a)	(b)	(c)
1	0		
2	1	0	
3	1	1	
4	1	0	
5	0		3
6	1	1	

NOT EXECUTED.

FOUND OK.

FOUND NO GOOD.

DETERMINED MEANINGLESS
FROM THE RESULT OF
STEP 3.

FIG. 9

STEP No.	CONTENT DESCRIPTION CHARACTER STRING	EXECUTION PROGRAM ADDRESS	POINT ALLOTMENT PROGRAM ADDRESS
1	PIF board PIO R/W check	xxxx	xxxx
2	PR(x) PIO r/w check	xxxx	xxxx
3	PIF read data check	xxxx	xxxx
4			

FIG. 10

GRADE STEPS & MESSAGES		
AREA 11 (PIF)	45	1-100, 3-0, 4-52
AREA 12 (PR(1))	0	
AREA 13 (PR(2))	5	10-120
AREA 81 (power supply)	0	
AREA 82 (cooling fan)	0	
AREA 83 (main mother)	0	
AREA 83 (sub mother)	0	
AREA 91 (EE_all)	----	
AREA 92 (PB_all)	----	
AREA 93 (REC_all)	----	
AREA 93 (sub mother)	----	

(X)

(Y)

(Z)

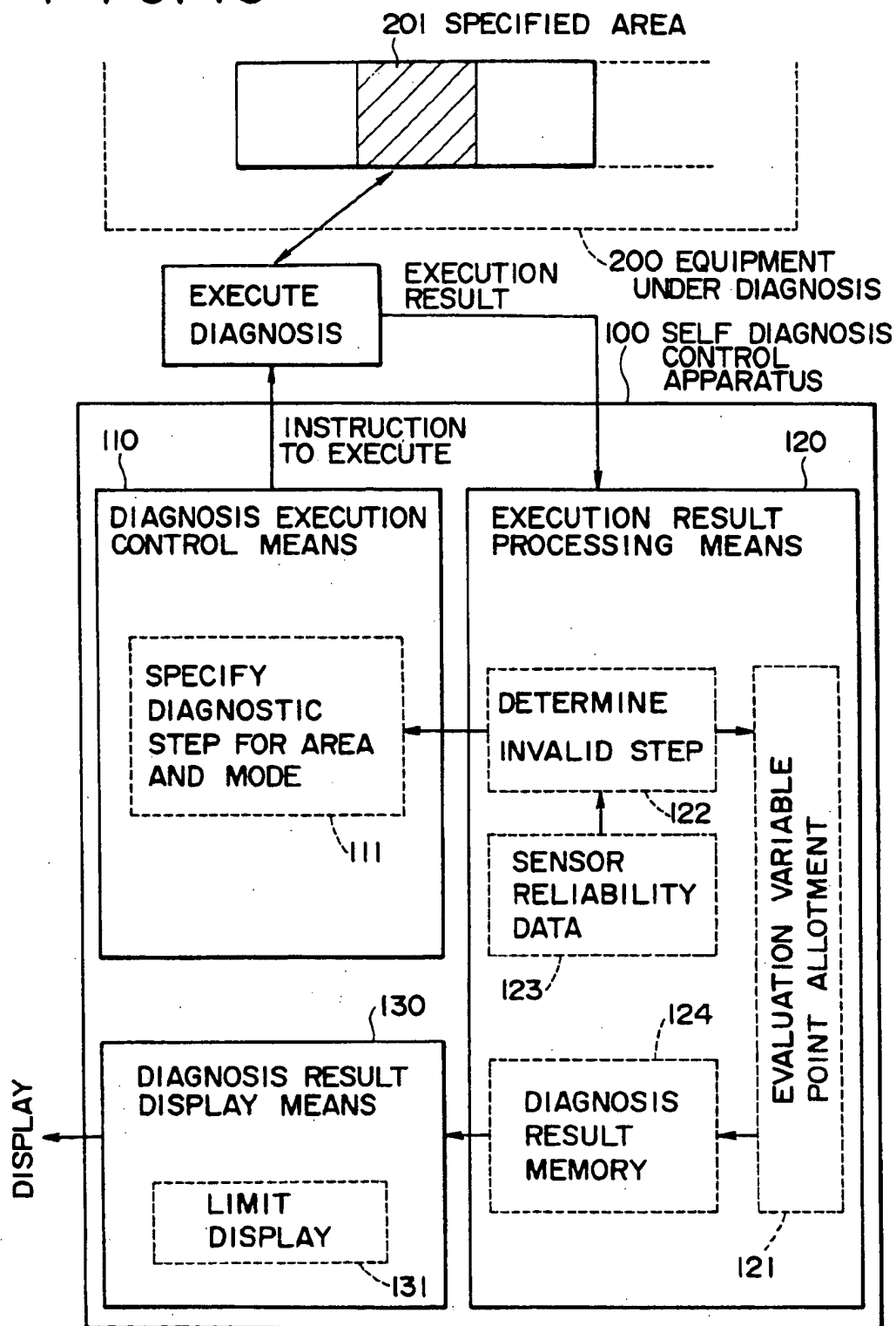
FIG. 11

MESSAGE No.	MESSAGE
1	PIF board may be defective
2	RF-REC block may be defective
3	Thread block may be defective
4	PB error rate is too high
5	
.	
.	
.	

FIG. 12

SENSOR No.	1	2	3	4				
RELIABILITY			1					

FIG. 13



METHOD AND CONTROL APPARATUS FOR SELF DIAGNOSIS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method and a control apparatus for self diagnosis and, more particularly, to a method and a control apparatus for self diagnosis suitable for use in such information processing apparatus as a data recorder having a mechanical portion.

2. Description of the Prior Art

It is known that some information processing, instrumentation, or control apparatus or systems have a self diagnostic capability. The self diagnostic capability of a data recorder, for example, includes an initial test to be performed when the data recorder is powered on, error indication, alarming, and error data storage in a regular operation, and various tests in a maintenance operation, among others.

Self diagnosis as generally referred to points at the diagnostic operations to be performed in the regular run of the apparatus as mentioned above, with emphasis placed on error detection. It sometimes can happen that one of the apparatus's functional portions which is used only once in a long while, one year for example, is found defective just before use. With a conventional data recorder, for example, it is a general practice to provide no capability of checking the entire recorder system including its infrequently used functional portion for a failure or no dedicated self diagnostic mode. However, in the field of instrumentation, some of apparatus including an oscilloscope are beginning to employ the capability for self-diagnosing their electric or electronic circuitry.

However, a limitation with the above-mentioned practice as performed on some instruments is that self diagnosis is adapted to apply only to electric or electronic portions of each instrument. Therefore, the practice cannot apply to any apparatus that contain mechanical portions to be subjected to self diagnosis. Such apparatus include a data recorder comprising a video tape recorder, for example.

Unlike an electric or electronic circuit, a mechanical portion of an apparatus cannot be definitely defined in terms of input and output. With the electric or electronic circuit, it is general practice to enter a test signal in it and observe a resultant output for diagnosis. Mechanically, it is not that simple; diagnosis must often be performed on multiple complicatedly interlaced portions.

At a laboratory level, a troubleshooting expert system is under development for inferring from a symptom a failing location based on an artificial intelligence (AI) approach. Implementing such expert system requires the use of a powerful computer system such as a workstation. Such implementation is unrealistic in terms of cost for the present invention which intends to incorporate the self diagnostic capability in an apparatus. In addition, the construction of an expert system requires a large-scale database on which inference must be made from a symptom as wisely as a human expert. However, it is often difficult to prepare and install such a database on an apparatus under fabrication in which a self diagnostic capability is incorporated.

Thus, for the self diagnosis of such apparatus as a data recorder, a system must be constructed which, still supporting a conventional diagnostic capability for error detection, provides a positive troubleshooting capability such as a dedicated self diagnostic mode. Further, the above-men-

tioned diagnostic system must be able to cope with such a complicated symptomatic situation for diagnosing a mechanical portion of the apparatus as that, while checking portion A, an abnormal condition has been unexpectedly detected in portion B. It is also desirable for the built-in diagnostic system to be implemented by a simple approach rather than complicated one typically supported by AI.

OBJECTS AND SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a simple self diagnostic method which can positively self-diagnose apparatus having complicatedly interlaced mechanical portions, and a controller for implementing the self diagnostic method.

In carrying out the invention and according to one aspect thereof, there is provided a self diagnostic method comprising the steps of dividing an apparatus under diagnosis into two or more areas subject to diagnosis, providing each of the areas with an evaluation variable, performing diagnosis in units of diagnostic steps classified by the area, allotting points to the evaluation variable of each area under diagnosis associated with the executed diagnostic step according to an execution result, and diagnosing each area based on a value of the evaluation variable obtained by adding up the allotted points.

When executing a diagnostic operation, it is arranged so that no step determined invalid by an execution result of another diagnostic step will be executed and an apparatus under diagnosis includes apparatus having a mechanical portion.

In carrying out the invention and according to another aspect thereof, there is provided a controller 100 as schematically shown in FIG. 13 comprising diagnosis execution control means 110, execution result processing means 120, and diagnostic result display means 130, wherein diagnosis is performed on a specified area 201 of an apparatus under diagnosis divided into multiple areas under diagnosis each provided with an evaluation variable, and a diagnostic result is determined upon receiving an execution result. Actually, as shown in FIG. 13, there is provided, between the controller 100 and the specified area 201, diagnosis executing means for executing diagnosis based on an instruction given by the diagnosis execution control means 110 and detecting the execution result to send it to the execution result processing means 120. However, the description of this diagnosis executing means is omitted herein because its specific configuration and implementation technique do not matter in carrying out the present invention.

The diagnosis execution means 110 has a diagnostic step specification capability 111 and executes diagnosis by specifying a diagnostic step for each area of the apparatus under diagnosis 200 and, further, for each diagnostic mode. In execution, it is arranged so that a determination by an invalid step decision capability 122 in the execution result processing means 120 is referenced not to execute diagnostic step determined invalid by an execution result of another step. The invalid step includes those steps which are determined invalid by sensor reliability data 123 obtained from the execution result of each step carried out by the execution result processing means 120 for example.

The execution result processing means 120 has an evaluation variable point allotting capability 121 to allot points to the evaluation variable of an associated area under diagnosis based on the execution result of each step and store a

diagnostic result consisting of and integrated value of evaluation variables (124). In this operation, if a step determined invalid by the decision of the invalid step decision capability 122 has already been executed, the evaluation variable point allotting capability 121 subtracts the allotted points from the associated area under diagnosis.

The diagnostic result display means 130 displays the diagnostic result obtained by the execution result processing means 120, starting with an area having a greatest evaluation variable. It also has display limit capability 131 for displaying only areas having evaluation variables above a certain level which can be varied.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating an entire configuration of the data recorder according to the present invention;

FIG. 2 is a block diagram illustrating an internal configuration of a CPU board 10 of FIG. 1;

FIG. 3 illustrates positions of self diagnostic modes in the present invention;

FIG. 4 illustrates an example of how an apparatus under diagnosis is divided into areas under diagnosis;

FIG. 5 illustrates how to operate self diagnostic modes;

FIG. 6-B is an overall flowchart illustrating self diagnostic mode operations;

FIG. 7 is a table for determining, from modes and areas under diagnosis, which diagnostic step is to be executed;

FIG. 8 is a table in which an execution result of each diagnostic step is entered;

FIG. 9 is a table for determining an address of a diagnostic step execution program and an address of a point allotment calculation program from an internal diagnostic step number;

FIG. 10 is a table in which a grade for each area under diagnosis is stored;

FIG. 11 is a table from which a message is obtained according to a message number;

FIG. 12 is a table for managing sensor output reliability; and;

FIG. 13 is a block diagram schematically illustrating the self diagnosis controller according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

A self diagnosis method and its controller practiced as a preferred embodiment of this invention as applied to a data recorder will be described in detail by way of example with reference to the accompanying drawings.

FIG. 1 is the block diagram illustrating the entire configuration of the data recorder practiced as the preferred embodiment of this invention. In the figure, each block does not indicate an actual board configuration; it indicates a functional block for self diagnosis, or an area under diagnosis.

Reference numeral 1 indicates data input/output interface which provides synchronization in data between a digital signal processor 2 and data I/O 20 and 21 of the data recorder. The digital signal processor 2 performs concurrent processing in four units for example to increase processing speed.

Digital data 20 entered is fed through the data input/output interface 1 to the digital signal processor 2 to be added with an error correction code. A resultant signal is fed to an RF-REC (record) system 3 to be recording-compensated for recording by a rotary head 4. In reproduction, this operational flow is reversed; however, instead of the RF-REC 3, an RF-PB (playback) system 5 is used.

A CPU board 10 is a CPU functional portion which controls the data recorder in its entirety. An outside remote control signal 24 including a self diagnosis start instruction and an inquiry for a diagnostic result is entered in the CPU board 10 through an interface board 25. A CP 26 is a control panel consisting of a keyboard and a display.

A mechanical sensor 12 represents all sensors installed in a tape travel system and a cassette compartment. Many of these sensors are optical sensors based on pairs of a light emitting diodes and phototransistors. A signal coming from the sensor 12 is processed by the CPU board 10 to be fed to a motor driver 11 as a control signal, driving motors and plungers for operating mechanical portions 13 through 19 of the tape travel system and the like shown.

FIG. 2 is the block diagram illustrating an internal configuration of the CPU board 10. It contains a main CPU 34 and three sub CPUs; IF CPU 31 for input/output control, PR CPU 32 for signal processing control, and SV CPU 33 for driver control. Controlled by the main CPU 34, each sub CPU in turn controls a corresponding portion to perform a data recorder operation.

The main CPU 34 also executes a self diagnostic sequence and contains a self diagnosis controller 38 for that purpose. The main CPU 34 is also provided with a storage unit 36 for storing a self diagnostic result and a data recorder state to be referenced before starting self diagnosis, a backup power supply for the storage unit 36, a reset line controller 41 for hardware-resetting the main CPU itself upon termination of the self diagnosis, and a power-on reset circuit 42 for controlling a reset operation at power-on.

In what follows, the self diagnosis method according to the invention and capabilities necessary for implementing it will be described.

First, the position of the self diagnosis mode according to the invention is described. As shown in FIG. 3 and as described above, the data recorder is provided with various self diagnostic capabilities including an initial test, a regular operation diagnostic capability, a test mode, and the self diagnosis mode positioned as a dedicated self diagnostic mode. The dedicated self diagnostic mode as referred to herein is a capability to be intentionally activated for checking the data recorder for any abnormal conditions. For example, this mode is activated, upon switching on the data recorder for starting a task, to check the data recorder for any trouble or troubleshoot the data recorder when it is found malfunctioning. Once in the self diagnostic mode, no regular operations including data recording/reproducing can be performed.

In the self diagnostic mode, the data recorder is divided into multiple areas to be subjected to diagnosis (FIG. 4). The self diagnostic mode comprises following four modes which respectively cover the areas under diagnosis. In mode 1, diagnosis is performed with no tape cassette loaded as with an independent reel rotational test. In mode 2, diagnosis is performed with an electric system calibration tape loaded as with a reproduction of a test signal tape. In mode 3, diagnosis is performed with a mechanical calibration tape loaded as with a tracking test and a head position control test. In mode 4, diagnosis is performed with a blank tape

loaded to check whether recording is made normally or not. One or more diagnostic steps (FIG. 9) are allocated as required to each area to be diagnosed in each mode and for execution.

It should be noted that the present embodiment can also be constituted so that self diagnosis is performed while a regular operation such as data recording/reproducing operation is in progress.

Now, referring to FIG. 4, the data recorder is divided into areas under diagnosis by way of example. To be specific, the data recorder is divided into seven divisions of a signal processing system, an audio signal processing system, a control system, a tape travel system, a tape loading system, a miscellaneous system, and an overall system. Each division is further divided into multiple functional blocks or areas suitable for particular diagnoses. Each area under diagnosis is assigned with an area number.

It should be noted that the areas in the miscellaneous division having area numbers on the order of 80 cannot be tested alone and the areas in the overall division having area numbers on the order of 90 have no specific object to be diagnosed. With the areas having numbers on the order of 90, the data recorder is operated in the same manner as a regular operation to make an overall diagnosis based on status variables such as voltages and on errors detected.

Operations of the self diagnostic mode and an operational flow of diagnosis will be described as follows. FIG. 5 indicates how to operate the self diagnostic mode, in which the mode is executed following procedures (1) through (8). In procedure (4), a self diagnosis starts. In procedure (5), diagnostic tapes are exchanged for changing modes as instructed by a diagnostic program halfway in its execution. FIG. 6 is an overall flow of the self diagnosis indicating the self diagnostic operations. In what follows, these self diagnostic operations will be described by referring to the overall flow.

First, in procedures (1) and (2), preparations necessary for carrying out the self diagnosis are made and desired self diagnostic mode and area under diagnosis are entered through the control panel or a remote control device. At this stage, the entered mode and area are only stored in the storage unit and therefore the mode is not yet active. When the self diagnostic mode is activated according to procedure (4) shown in FIG. 5, the self diagnosis starts following procedures (3) through (5) shown in FIG. 6.

Each diagnostic step, in procedure (6), obtains the number of a step to be executed next from a current setting by referring to ROM data in which diagnostic steps to be executed for each mode have been set for each area as shown in FIG. 7. Meanwhile, in procedure (8), contents of RAM in which execution results of diagnostic steps are stored as shown in FIG. 8 are referred to so that, if a specific diagnostic step is found to have already been executed, the step will not be executed, or if a diagnostic step to be executed next based on the result of another diagnostic step is found invalid, the step will not be executed, thereby passing control to a next diagnostic step.

Especially, if the output of a sensor is found unreliable by the result of a diagnostic step for example, the execution results of the diagnostic steps associated with the sensor output will subsequently be handled as being unreliable. Consequently, a table listing the reliability data of each sensor is prepared beforehand as shown in FIG. 12. The table is referred to every time a diagnostic step is to be executed to determine whether the step is valid or invalid.

When changing a diagnostic mode to another, if a diag-

nostic tape need be changed to another, a current tape is ejected and a required tape is inserted, a tape exchange event is displayed, and the operational flow is held until the tape has been fully loaded as in procedures (9) and (10). Each step is executed by obtaining an address of a program to be executed from its step number based on the ROM data which specifies the preset execution contents of each step as shown in FIG. 9.

If a so-called system error occurs during execution of a diagnostic step, the error will be handled as follows in procedures (12) through (15). The system error is one of the following four types: (1) an error whose occurrence is self evident from the nature of self diagnosis and which may be ignored; (2) an error whose occurrence is expected and its presence or absence is positively used for self diagnosis; (3) an unexpected error which is useful for self diagnosis and allows it to proceed; and (4) an unexpected error which does not allow self diagnosis to proceed.

This error handling is performed in each diagnostic step, in which a type of detected error is determined to perform necessary processing accordingly. That is, the error of type (1) mentioned above is ignored in procedure (13), the error of type (4) causes the diagnosis to terminate in procedure (14), and the errors of types (2) and (3) add up a grade value of an suspected area under diagnosis depending on their contents in procedure (15). The grade (or an evaluation value, also called an evaluation variable herein) as used herein indicates a degree of failure in an associated area under diagnosis. Its value is stored for each area under diagnosis and may be 0 to 255. This value increments in units of 1 to 10 depending on an execution result of each step. As this value increases, it is more probable that the area under diagnosis is failing.

When a diagnostic step has terminated in procedure (17), the grade value for an area under diagnosis associated with that diagnostic step is manipulated; that is, point allotment is performed. Points are allotted by a point allotting program preset for each diagnostic step. As shown in FIG. 9, an address of each point allotting program can be referenced in ROM which relates a step number with a diagnosis execution program. For example, if executing a diagnostic step to test area 22 results in a no good condition, the grade of the area 22 is increased by 30 points. If, at this time, area 23 is also suspicious, its grade is increased by 20 points for example.

The grade value thus allotted to the area under diagnosis is stored in an execution result table (FIG. 10) to be described, along with information on the diagnostic step executed and diagnostic message. At the same time, whether a diagnostic step has been executed or not and, if executed, its result are entered in the table of FIG. 9 for each diagnostic step. If the execution result indicates that execution of another diagnostic step is meaningless, that step is flagged invalid. If the diagnostic step having an invalid flag has already been executed, it means that its execution will be meaningless, thus requiring to subtract the points allotted to the associated area from its grade value.

Some diagnostic steps forcibly change a state in the data recorder for testing. Once changed, it is often difficult to restore the original state even if the state is in memory. Therefore, if this happens, a hardware reset operation is executed in procedure (18). When specified diagnostic steps have all been executed or if it is determined that a fatal error detected in procedure (14) has made impossible the continuation of the diagnosis, the diagnosis is terminated (refer to (*1) in FIG. 6).

The diagnostic result is stored in the RAM as a grade value for each area under diagnosis as shown in FIG. 10; that is, points to be allotted as results of the execution of diagnostic steps are added up in the RAM. At the same time, a number of a diagnostic step determined failing and an associated diagnostic message are stored (steps & messages). For example, as shown in FIG. 11, all messages stored in the ROM beforehand are assigned with numbers which are stored in the RAM of FIG. 10. In FIG. 10, "1-100" indicates that step 1 is no good and message 100 must be referred to, "3-0" indicates that step 3 is no good and there is no message to be referred to, and so on.

Now, when the diagnosis has been completed, the result is displayed on the control panel starting with an area under diagnosis having a highest grade value, in a form "area xx is suspicious of failing," in procedure (20). In addition to this message, a special message informing a service engineer of the reason of failure determination may be displayed for each diagnostic step, for example.

It should be noted here that, as a general rule, no area under diagnosis having less than 16 points will be displayed. This is because, if areas under diagnosis slightly suspicious of failing are displayed in large quantities, practicality of the self diagnosis would be lost. However, the grade or the number of points may be lowered from 16 if a service engineer for example requires more detail failure information.

If an attempt is made to execute a test of the same mode twice, it is judged that a retry of the test is requested, automatically erasing previous data. The previous data may also be manually erased by an operator.

Data structures, or tables, to be referred to during execution of the above-mentioned diagnostic program will be described as follows. These tables contain two types of data; predefined data stored in the ROM and data stored in the RAM as a result of diagnostic execution.

A table of FIG. 7 is used to determine which step is to be executed based on a mode and an area under diagnosis specified on the control panel. This table is written in the ROM of the main CPU 34 beforehand. Each diagnostic step is represented in an internal step number uniquely set in the data recorder. In the table, a number or numbers indicated for each area under diagnosis are step numbers. For example, when all steps of area 11 are to be checked, steps 1, 3, and 4 are executed (*1); when all steps of mode 1 are to be checked, steps 1, 2, 7, and 8 are executed (*2).

FIG. 8 is a table in which an execution result of each diagnostic step is entered. This table is created in the RAM of the main CPU 34. Entries in this table include, for each diagnostic step, whether the step has been executed or not, a result of execution, and whether the result is invalid or valid. In the table, (a) indicates whether the step has been executed or not and is 1 if executed; (b) indicates the execution result and is 0 if found good; and (c) indicates that the execution result of the step is found invalid by another step, and is the number of the invalid step.

FIG. 9 is a table for obtaining an address of a step execution program from the internal step number and an address of a grade calculation program from a step execution result. This table is also written in the ROM beforehand. In the table, "Execution program address" column contains a start address of an execution program corresponding to a specified diagnostic step and "Point allotment program address" column contains an address of the point allotment program for recalculating the grade of the diagnostic step from a result of a subsequent diagnostic step. "Content

description character string" column contains information to be used when displaying the content of the specified diagnostic step on the control panel.

FIG. 10 is a table to store a grade for each area under diagnosis and is stored in the RAM. In addition to the grade, this table stores an internal number of a diagnostic step which manipulated the grade and an internal number of a message given by the step. In the figure, (X) indicates an area under diagnosis for which a program is executed as a main target; (Y) indicates an area under diagnosis circumstantially suspected of failure from a program execution result; and (Z) indicates an area under diagnosis for which an overall diagnosis is performed without specifying a particular location. It should be noted that the (Z) area under diagnosis indicates no particular area under diagnosis and therefore has no grade value. "Steps & messages" column contains the number of a failed step and a storage number (refer to FIG. 11) of an associated message. For example, "1-100" means that step 1 is no good and message number 100 must be referred to for description; "3-0" means that step 3 is no good and there is no corresponding message.

FIG. 11 is a table for obtaining a message from a message number and is also stored in the ROM beforehand.

FIG. 12 is a table for managing sensor output reliability. If a result of a diagnostic step indicates that an output of a sensor is not reliable, reliability "1" is recorded for the sensor. This table is referenced by each diagnostic step to determine whether its execution result is valid or invalid.

As described and according to the invention, actions necessary for self diagnosis are taken automatically in general, realizing faster and more accurate self diagnosis than prior-art techniques. A self diagnosis is performed by selecting from multiple diagnostic steps necessary ones. More steps executed will result in a more accurate result. If required, only a minimal number of steps may be executed for a meaningful result.

If an execution result of each diagnostic step indicates two or more suspicious areas under diagnosis, the result can be used for evaluating such areas by manipulating their grade values.

If an error has been detected during the execution of a diagnosis, manipulating the grade values of multiple associated areas under diagnosis allows to positively use the error for the diagnosis.

Further, a failure which cannot be located by a single test can be checked by using circumstantial evidence obtained by other diagnostic steps without resorting to complex inference. In addition, introduction of grade values for evaluation allows to use a test which puts out a vague result. While the preferred embodiment of the invention has been described using specific terms, such description is for illustrative purpose only, and it is to be understood that changes and variations may be made without departing from the spirit or scope of the appended claims.

What is claimed is:

1. A method of self diagnosing an apparatus having a plurality of functional areas, the method comprising the steps of:

performing a plurality of diagnostic steps on one of said functional areas of said apparatus, said one functional area having a grade value representing its functional operativeness, each of said plurality of diagnostic steps producing a respective diagnostic result;

allocating a respective point value to each said respective diagnostic result; and

varying said grade value of said one functional area by

9

- said respective point value of each said respective diagnostic result.
2. The method of claim 1, wherein the step of performing a plurality of diagnostic steps includes the steps of:
- storing validity data for each of said diagnostic steps based upon diagnostic results of previously performed diagnostic steps;
 - determining whether a respective diagnostic step to be performed is a valid step based upon the stored validity data;
 - performing said respective diagnostic step if it is determined to be a valid step;
 - determining whether other ones of said diagnostic steps are invalid steps based upon the produced diagnostic result of said performed diagnostic step; and
 - altering said stored validity data to indicate the invalidity of said other ones of said diagnostic steps that are determined to be invalid.
3. The method of claim 2, wherein the step of performing a plurality of diagnostic steps further includes the step of adjusting said grade value in accordance with said point values of said diagnostic results of previously performed diagnostic steps that are determined to be invalid steps so as to nullify previous affects on said grade value by diagnostic steps that are determined to be invalid.
4. The method of claim 3, wherein the step of varying said grade value is carried out by increasing said grade value by said point value of said diagnostic result of each said performed diagnostic step; and said step of adjusting said grade value is carried out by reducing said grade value by said point value of said diagnostic result of each said previously performed diagnostic step that is determined to be an invalid step.
5. The method of claim 1, wherein said diagnostic steps are performed on a plurality of functional areas of said apparatus, each of said functional areas having a respective grade value representing a functional operativeness of said respective functional area.
6. The method of claim 5, further comprising the step of displaying said functional areas in order of decreasing grade values.
7. The method of claim 5, further comprising the step of displaying said functional areas having respective grade values greater than a predetermined value.
8. The method of claim 1, wherein the step of allocating a respective point value is carried out by retrieving from a memory a value corresponding to said respective result for said respective diagnostic step and assigning the retrieved value as said respective point value.
9. A diagnosis system for diagnosing an apparatus having a plurality of functional areas, the diagnosis system comprising:
- means for performing a plurality of diagnostic steps on one of said functional areas of said apparatus under diagnosis, said one functional area having a grade value representing its functional operativeness, each of said plurality of diagnostic steps producing a respective diagnostic result;
 - means for allocating a respective point value to each said respective diagnostic result; and
 - means for varying said grade value of said one functional area by said respective point value of each said respective diagnostic result.
10. The diagnosis system of claim 9, wherein said means for performing a plurality of diagnostic steps includes:
- means for storing validity data for each of said diagnostic

10

- steps based upon diagnostic results of previously performed diagnostic steps;
 - means for determining whether a respective diagnostic step to be performed is a valid step based upon the stored validity data;
 - means for performing said respective diagnostic step if it is determined to be a valid step;
 - means for determining whether other ones of said diagnostic steps are invalid steps based upon the produced diagnostic result of said performed diagnostic step; and
 - means for altering said stored validity data to indicate the invalidity of said other ones of said diagnostic steps that are determined to be invalid.
11. The diagnosis system of claim 10, wherein said means for performing a plurality of diagnostic steps further includes means for adjusting said grade value in accordance with said point values of said diagnostic results of previously performed diagnostic steps that are determined to be invalid steps so as to nullify previous affects on said grade value by diagnostic steps that are determined to be invalid.
12. The diagnosis system of claim 11, wherein said means for varying said grade value includes means for increasing said grade value by said point value of said diagnostic result of each said performed diagnostic step; and said means for adjusting said grade value includes means for reducing said grade value by said point value of said diagnostic result of each said previously performed diagnostic step that is determined to be an invalid step.
13. The diagnosis system of claim 9, wherein said means for performing a plurality of diagnostic steps is operable to perform diagnostic steps on a plurality of functional areas of said apparatus, each of said functional areas having a respective grade value representing a functional operativeness of said respective functional area.
14. The diagnosis system of claim 13, further comprising means for displaying said functional areas in order of decreasing grade values.
15. The diagnosis system of claim 13, further comprising means for displaying said functional areas having respective grade values greater than a predetermined value.
16. The diagnosis system of claim 13, further comprising means for designating said plurality of functional areas of said apparatus to be diagnosed.
17. The diagnosis system of claim 13, further comprising storage means for storing said diagnostic steps to be performed to diagnose each of said functional areas of said apparatus; and wherein said means for performing further includes means for retrieving from said storage means said diagnostic steps to be performed.
18. The diagnosis system of claim 17, wherein said apparatus is a data recording device; and said storage means stores diagnostic steps to be performed to diagnose each of said functional areas in a plurality of self diagnostic modes, said plurality of self diagnostic modes including an independent reel rotation diagnostic mode, an electric system calibration diagnostic mode, a mechanical calibration diagnostic mode, and a data recording mode.
19. The diagnosis system of claim 9, further comprising storage means for storing values representing point values for each possible diagnostic result of each of said diagnostic steps; and wherein said means for allocating a respective point value includes means for retrieving from said storage means a value corresponding to said respective result for said respective diagnostic step and means for assigning the retrieved value as said respective point value.
20. A self diagnosis control apparatus which divides an apparatus under diagnosis into a plurality of areas under

11

diagnosis, each provided with an evaluation variable for performing a diagnosis in units of said areas under diagnosis, comprising:

- (1) self diagnosis execution control means for executing the diagnosis based on diagnostic steps corresponding to said areas under diagnosis, at least some of said diagnostic steps corresponding to a plurality of said areas under diagnosis, said self diagnosis execution control means having a first storage means for storing said diagnostic steps corresponding to each of said areas under diagnosis from which specified diagnostic steps are read for execution and a storage means for storing information which tells whether any of said diagnostic steps has been executed;
- (2) execution result processing means for adding up points allotted to said evaluation variable based on an execution result of each diagnostic step, said execution

12

result processing means controlling detecting means for detecting whether there is any diagnostic step that is determined by another diagnostic step to be invalid based on its diagnostic result and said diagnosis execution control means so that no diagnostic step determined invalid based on a detection result is to be executed, wherein, if an already executed diagnostic step is determined invalid based on said detection result, no point allotted by said invalid step is added to said evaluation variable;

- (3) display means for displaying a state of said apparatus under diagnosis based on a value of said evaluation variable for each of said areas; and
- (4) specification means for specifying said areas under diagnosis.

* * * * *